TOWARDS ROBUST CONVERSATIONAL SPEECH RECOGNITION AND UNDERSTANDING

A Dissertation
Presented to
The Academic Faculty

By

Chao Weng

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
December, 2014

TOWARDS ROBUST CONVERSATIONAL SPEECH RECOGNITION AND UNDERSTANDING

Approved by:

Dr. Chin-Hui Lee, Committee Chair
Professor, School of ECE
Georgia Institute of Technology

Dr. Elliot Moore II
Associate Professor, School of ECE
Georgia Institute of Technology

Dr. Biing-Hwang (Fred) Juang, Advisor
Professor, School of ECE
Georgia Institute of Technology

Dr. Yajun Mei
Associate Professor, School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Mark Clements
Professor, School of ECE
Georgia Institute of Technology

Date Approved: July 17, 2014

To my parents and my wife,

for their boundless

love and support

ACKNOWLEDGMENT

I would like to express the deepest gratitude to my advisor, Prof. Biing-Hwang (Fred) Juang. Benefited from his excellent teaching and guidance these years, I have gained a great deal of expertise and developed a deep passion in speech recognition from knowing nothing about it. He has been and will always be a role model for my career.

Thanks to the professors who spend their precious time to serve on my dissertation committee: Prof. Chin-Hui Lee, Prof. Mark Clements, Prof. Elliot Moore II and Prof. Yajun Mei. Special thanks go to Prof. Chin-Hui Lee for the collaborations with his group.

During the pursuit of my PhD at Georgia Tech these years, I owe a debt of gratitude to many people here. Thanks to our group members from whom I learned a great deal: Umair Altaf, Mingyu Chen, Qiang Fu, Dwi Sianto Mansjur, Zhong Meng, Antonio Moreno, Sung-Hwan Shin, Mehrez Souden, Ted Wada, Jason Wung and Yong Zhao; Thanks to I-fan Chen, You-Chi Cheng, Zhen Huang, Kehuang Li for the valuable discussions and collaborations; Special thanks to Prof. Guotong Zhou for managing the Georgia Tech Shanghai program and the friends from this program for sharing many laughs; Thanks to Pat Dixon, Jennifer Lunsford, Daniela Staiculescu, and Tasha Torrence for their great administrative support.

I am also very fortunate to meet and work with groups of excellent researchers and friends outside Georgia Tech: Daniel Povey at Johns Hopkins University; David Thomson, Patrick Haffner and Diamantino Caseiro at AT&T Labs Research; Dong Yu, Mike L. Seltzer and Jasha Droppo at Microsoft Research; Shinji Watanabe at Mitsubishi Electric Research Laboratories.

Finally, I would like to thank my parents, my parents-in-law and my wife, Tianyi, for their immeasurable love and support during my PhD study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Challenges

While significant progress has been made in automatic speech recognition (ASR) during the last few decades, recognizing and understanding unconstrained conversational speech remains a challenging problem in the field. Unlike read or highly constrained speech, spontaneous conversational speech is often ungrammatical and ill-structured. One can expect over 90% word accuracy on a large vocabulary continuous speech recognition (LVCSR) task with speech that is rendered in a dictation speaking style, but this accuracy would be dramatically degraded for a spontaneous conversational task, *e.g.*, the Switchboard [3]. From the benchmarks of ASR performance in word error rates (WERs) for DARPA-sponsored tasks [4, 5, 6] which is shown in Fig.1, we can tell that for the read speech tasks, *e.g.*, RM 1K, WSJ 5K and 20K tasks, one already can limit the WERs below 10%. However, the WERs are still relatively high for those spontaneous conversational speech tasks.

Another challenge in recognizing and understanding the conversational speech comes from the adverse acoustical environments which usually degrade the system performance substantially. On the one hand, in an adverse acoustical environment, the clean conversational speech is corrupted by additive noise and channel distortions. On the other hand, the presence of additive noise can sometimes change the way the speaker speaks which is known as the Lombard effect [7]. All these factors lead to a severe mismatch between training and testing conditions. A system does not degrade very much under the mismatched conditions is called a robust ASR system. Recognizing conversational speech in the presence of a competing talker, *a.k.a* the cocktail party problem, is still one of the unsolved problems in the field of robust ASR.

Compared to speech recognition, the problem of speech understanding can be even more complicated. In this thesis, we focus our attention on extracting the semantic notions

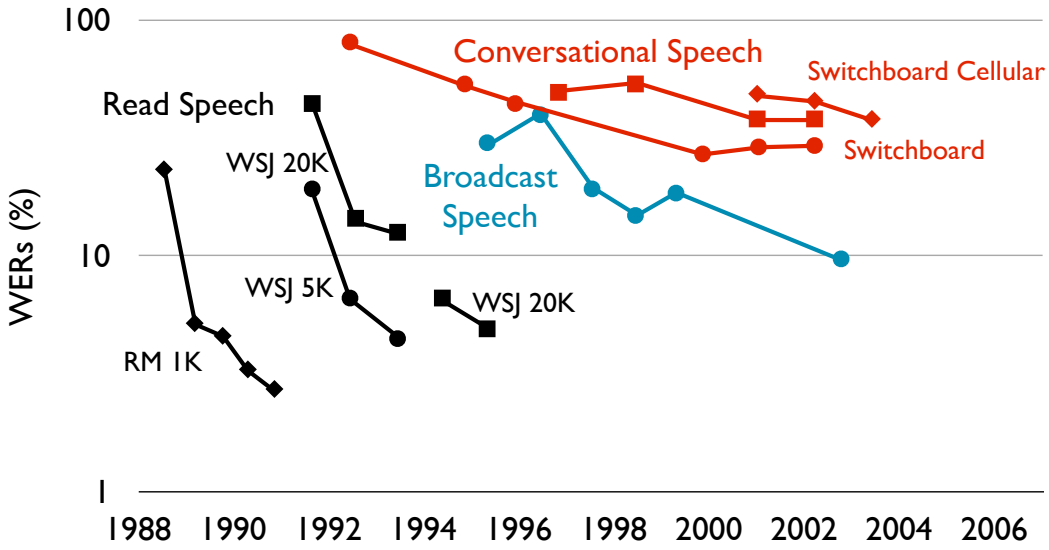

Figure 1: Benchmarks of ASR performance in WERs on DARPA-sponsored tasks

from a conversational speech. As mentioned earlier, the WERs on typical conversational ASR tasks are relatively high making the linguistic features extracted from the ASR outputs very unreliable. Furthermore, extracting semantic notions from conversational speech has a fundamental difficulty due to the often excessive amount of ill-forms such as partial or repetitive words, partial sentences, and non-linguistic mummers, in spontaneous conversations.

1.2 Background of Speech Recognition

1.2.1 Key Components of a Speech Recognition System

In general, the key components of a speech recognition system are feature extraction, acoustic models, pronunciation models, language models and decoders. Speech recognition can be concisely expressed in the equation,

$$W^* = arg\max_W P(W|X) = arg\max_W \frac{P(X|W)^\kappa P(W)}{\sum_{W'} P(X|W')^\kappa P(W')} = arg\max_W P(X|W)^\kappa P(W), \quad (1)$$

where X is the sequence of input feature vectors generated by the feature extraction component; $P(X|W)$ is the acoustic model and κ is the scaling factor for the acoustic score; and $P(W)$ is modeled by the combined pronunciation and language models. The best word sequence W^* is found by the decoders.

1.2.1.1 Feature Extraction

The speech signal can be considered as the output of a slowly time-varying linear system driven by an excitation signal. The excitation signal has the form of a quasi-periodic glottal wave for the voiced speech and the form of random noise for the unvoiced speech. The linear system represents the vocal tract which changes the spectrum of the speech coming out of the lips. Spectral shape is the most important feature representation for ASR which may be augmented by some auxiliary features to further improve the performance, *e.g.*, pitch information for the tonal languages. A common choice of spectral shape parameters is *cepstrum*, which is defined as the inverse Fourier transform (IFT) of the log-spectrum. It is a *homomorphic* transform that allows us to separate the source (excitation) and system response (tract information),

$$x[n] = e[n] * h(n) \quad \rightarrow \quad \hat{x}[n] = \hat{e}[n] + \hat{h}[n]. \tag{2}$$

A good feature extraction algorithm for an ASR system also needs to take into accounts the perceptual factors and the convenience for statistical modeling. Mel-Frequency Cepstrum Coefficients (MFCCs) [8] is the most widely used cepstral feature in ASR where a filter-bank with a nonlinear frequency scale is used to approximate the behavior of a human auditory system. Denote by $H_m[k]$ the mth triangular filter in the filter-bank, and $X[k]$ denotes the STFT of the input signals, we compute the log-energy at the output of each filter as,

$$S[m] = \log\left[\sum_{k=0}^{N-1} |X[k]|^2 H_m[k]\right], \quad 0 \le m < M, \tag{3}$$

the MFCCs are the DCT of these M filter outputs,

$$c[n] = \sum_{m=0}^{M-1} S[m] \cos(\pi n(m + 1/2)/M), \quad 0 \le n < M. \tag{4}$$

The use of Euclidean distance as distortion measures in MFCCs implies they can be easily modeled by the GMMs. Additionally, the de-correlation effect of the DCT alleviates the limitation of diagonal covariance matrices often used in a GMM-HMM system which

makes the assumption that the feature dimensions are independent from each other. However, recent DNN-HMM systems appear to work better with the features without DCTs , *i.e.*, log Mel filter-bank features as in (3). Another often used cepstrum feature is perceptual linear prediction (PLP) [9], which approximates the spectral shape using an autoregressive all-pole model. In the PLP feature extraction, prior to the all-pole modeling, the spectrum will be warped in Bark scale, pre-emphasized by the simulated equal-loudness curve and cubic-root amplitude compressed which are beneficial for ASR in noisy conditions

1.2.1.2 Acoustic Models

The adoption of HMMs [10, 11] for acoustic modeling is one of the greatest breakthroughs in the field of ASR. In an LVCSR system, as shown in Fig.2, each context-dependent phoneme, *e.g* triphone or quinphone, is represented by an HMM. Speech frames are aligned to certain HMM states using the forward-backward or Viterbi algorithm. In a GMM-HMM system, the state emission is modeled by a mixture of Gaussians,

$$\log p(\mathbf{o}_t|s_j) = \log \sum_{m=1}^{M} \pi_{jm} \mathcal{N}_{jm}(\mathbf{o}_t|s_j), \tag{5}$$

where $\mathbf{o}_t$ is the feature vector at frame t and s_j denote the jth HMM state. To control the model complexity, multiple states will be tied together using the decision tree [12] with each node containing a set of questions about contexts, positions, tones, *etc.*

1.2.1.3 Pronunciation and Language Models

Pronunciation models translate a word to a sequence of phonemes. A good pronunciation model is crucial to an ASR system. In general, a pronunciation model is constructed using a knowledge-based lexicon which specifies the mapping relationships between each word and its phonetic transcription in the vocabulary. For those out-of-vocabulary words (OOVs), a grapheme-to-phoneme conversion needs to be conducted.

The task of language modeling is to assign a probability to an input word sequence. The most widely used language models (LMs) are n-gram language models which estimate

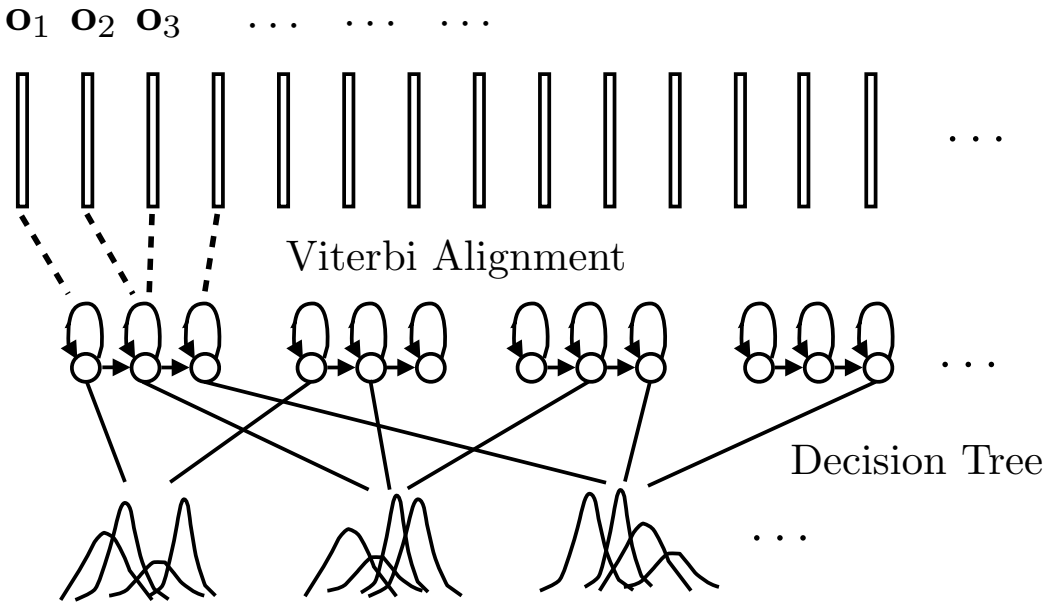

Figure 2: Acoustic Models in a GMM-HMM LVCSR system

the probability of certain word w given its history h using the MLE principle,

$$P(w|h) = \frac{n(w,h)}{\sum_{w'} n(w',h)}. \tag{6}$$

Smoothing techniques are essential in the construction of n-gram LMs. An LM without smoothing will assign zero probabilities to unseen words which will cause a disaster to ASR systems. Common smoothing techniques include Jelinek-Mercer interpolation [13], Katz backoff [14], Witten-Bell smoothing [15], absolute discounting and Kneser-Ney smoothing [16]. An empirical study of the smoothing techniques can be found in [17] which showed Kneser-Ney smoothing works best empirically.

Another type of popular language modeling is maximum entropy LMs [18, 19] which estimate the probability $p(w|h)$ using maximum entropy principle. Using the Lagrange multipliers for the constraints on the distribution $p(w|h)$, it can be easily proved that when the entropy of the distribution is maximized it will have the log-linear form,

$$P(w|h) = \frac{\exp(\sum_i \theta_i f_i(h,w))}{\sum_{w'} \exp(\sum_i \theta_i f_i(h,w'))}, \tag{7}$$

where $f_i(h,w)$ is the feature function. The advantage of the maximum entropy LMs is their flexibility to incorporate a rich set of features. The parameters θ_i are usually learned using

gradient descents or LBFGS [20]. Recently, neural network LMs are introduced in [21] and [22]. In fact, maximum entropy models can be viewed as neural network LMs with only an affine transform layer and a softmax nonlinear layer. The state-of-the-art performance has been reported using recurrent neural network LMs in [22].

1.2.1.4 Decoders

Taking into accounts all the models listed above, a decoder will search for the most likely word sequence as in (1) given the input speech feature vectors. Recent LVCSR decoders are mostly based on the weighted finite-state transducers (WFSTs) [23]. In a WFST based decoder framework, a static decoding graph HCLG is first constructed via the compositions of four WFSTs which encode the information of acoustic models, phonetic context, pronunciation and language models respectively (please find more technical details in Chapter 2 and 4). Note that the decoding graph is constructed independently of the decoder which leads to a more flexible decoder architecture.

The key algorithm of a decoder is token passing on the decoding graph with beam pruning. Each token is associated with one state in the HCLG graph and carries the cost up to the current frame and the traceback information. When one more speech frame is being processed, we pass all the active tokens through the corresponding arcs in the graph and accumulate the acoustic costs at the same time. The cost deviation from the present best path is called *beam-width*. All the tokens that fall outside the beam-width will be cut off. After the token passing is done, we can then perform a traceback to find the best sequence.

1.2.2 Discriminative Training for Speech Recognition

Acoustic models trained with MLE principle usually will not lead to a system that commits minimum number of recognition errors. The central idea of discriminative training for speech recognition is to formulate new criteria that directly link to the performance metric, *e.g.*, WER, and train the models according to the new objective functions. So far,

most widely used discriminative training (DT) methods include maximum mutual information (MMI) [24], minimum classification error (MCE) [25], minimum phone/word error (MPE/MWE) [26] and Boosted MMI [27].

MCE is the first discriminative training method that directly links the objective function to the recognition errors. Let X_r, $r = 1, \cdots, R$, be the utterances in the training set, W_r be the transcription word label for X_r. The discriminant function for a hypothesized word sequence W is defined as,

$$g(X_r, W) = \log\left[P(X_r|W)^\kappa P(W)\right], \tag{8}$$

The misclassification measure is thus,

$$d(X_r) = -g(X_r, W_r) + \log\left[\frac{1}{|W|}\sum_{W \neq W_r} \exp[g(X_r, W)]\eta\right]^{\frac{1}{\eta}}. \tag{9}$$

With proper smoothing using the sigmoid function, the objective function of MCE is formulated as,

$$\mathcal{L}(\theta) = \sum_{r=1}^{R} \ell(d(X_r)), \tag{10}$$

where $\ell(\cdot)$ is the sigmoid function. For the MMI or boosted MMI, the objective functions are,

$$\mathcal{F}_{\text{MMI}}(\theta) = \sum_{r=1}^{R} \log \frac{P(X_r|W_r)^\kappa P(W_r)}{\sum_W P(X_r|W)^\kappa P(W) e^{-b\mathcal{A}(W,W_r)}}, \tag{11}$$

where $\mathcal{A}(W, W_r)$ is the accuracy function which specifies the number of correct events at different levels. For the MPE/MWE, the objective functions are,

$$\mathcal{F}_{\text{MPE}}(\theta) = \sum_{r=1}^{R}\sum_{W} \mathcal{A}(W_r, W)\frac{P(X_r|W)^\kappa P(W)}{\sum_{W'} P(X_r|W')^\kappa P(W')}. \tag{12}$$

1.2.3 Deep Neural Networks for Speech Recognition

Recently, speech recognition has been made a great leap forward with the use of deep neural networks (DNNs). DNNs have applied to three components of a speech recognition system: feature extraction, acoustic models and language models.

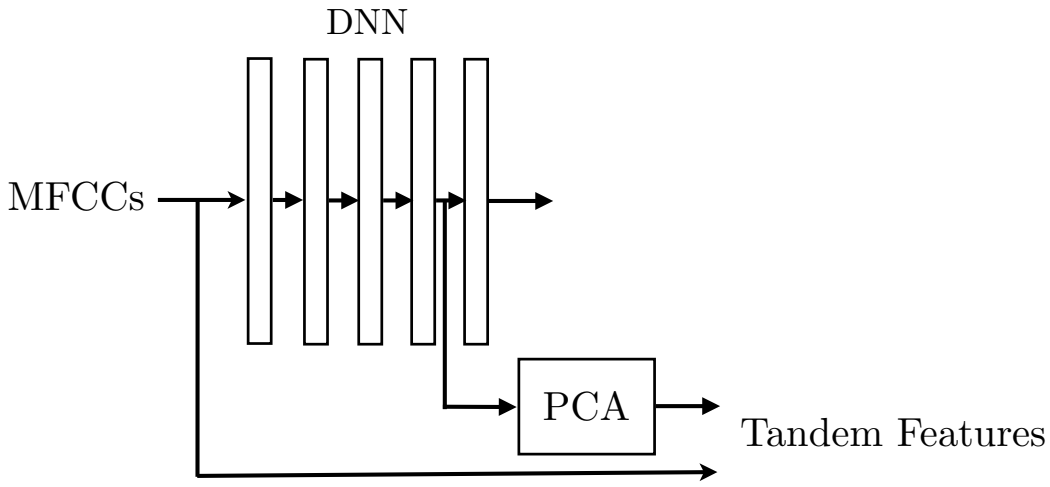

Figure 3: Tandem Features Extractions

The systems which use DNNs for the feature extraction are known as Tandem systems. In the original Tandem system [28], as shown in Fig.3, after a DNN is trained using the phoneme posteriors, the pre-nonlinearity outputs of the DNN will be decorrelated with PCA and then concatenated with the original MFCCs features. And this augmented features will be fed to the GMM-HMM system. Later on, some other types of Tandem systems are introduced. The probabilistic Tandem system concatenates the neural network output posteriors with original MFCCs features. The bottleneck Tandem system [29] first train a DNN with a bottleneck hidden layer in the middle and the linear output of the bottleneck layer is taken as output instead of the posteriors.

As shown in Fig.4, hybrid DNN-HMM systems [30] directly use DNNs to generate the acoustic score for each HMM state emission. So called pseudo log-likelihoods derived from the DNN outputs are used as the state emissions,

$$\log p(\mathbf{o}_t|s_j) \propto \log p(s_j|\mathbf{o}_t) - \log p(s_j), \tag{13}$$

where the state priors $\log p(s_j)$ can be estimated using the state alignments on the training speech data. The input features vectors x_t to the first layer of DNNs usually use a context of l frames [31], *e.g.*, $l = 9$ or $l = 11$.

As mentioned earlier, the neural networks have been used in the language modeling.

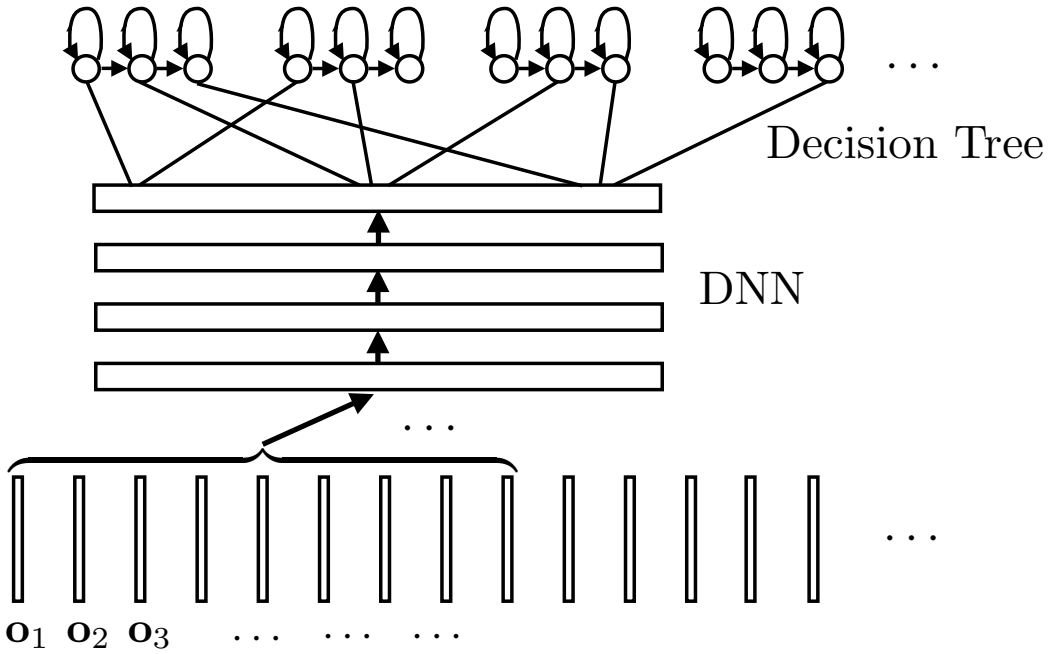

Figure 4: Acoustic Models in a hybrid DNN-HMM LVCSR system

Inspired by the fact that the deeper architecture improves acoustic modeling, some researchers [32] have used more hidden layers in neural network language modeling. More details can be found in Chapter 6.

1.3 Speech Recognition and Understanding in a WFST framework

Fig.5 shows the diagram of speech recognition and understanding in a WFST view. The feature extraction module converts the input speech signal into a series of feature frames. Each feature frame is represented by a HMM state according to the acoustic models. The state sequences are translated to phoneme sequences using the information provided by the HMM structure and the phonetic contexts. Pronunciation and language models together are used to determine the most likely word sequence. Finally with the semantic decoder, the semantic notions can be extracted from the given speech.

Towards a more robust conversational speech recognition and understanding system, in this thesis, we will focus on two components, acoustic models and semantic decoders which are highlighted in the Fig.5.

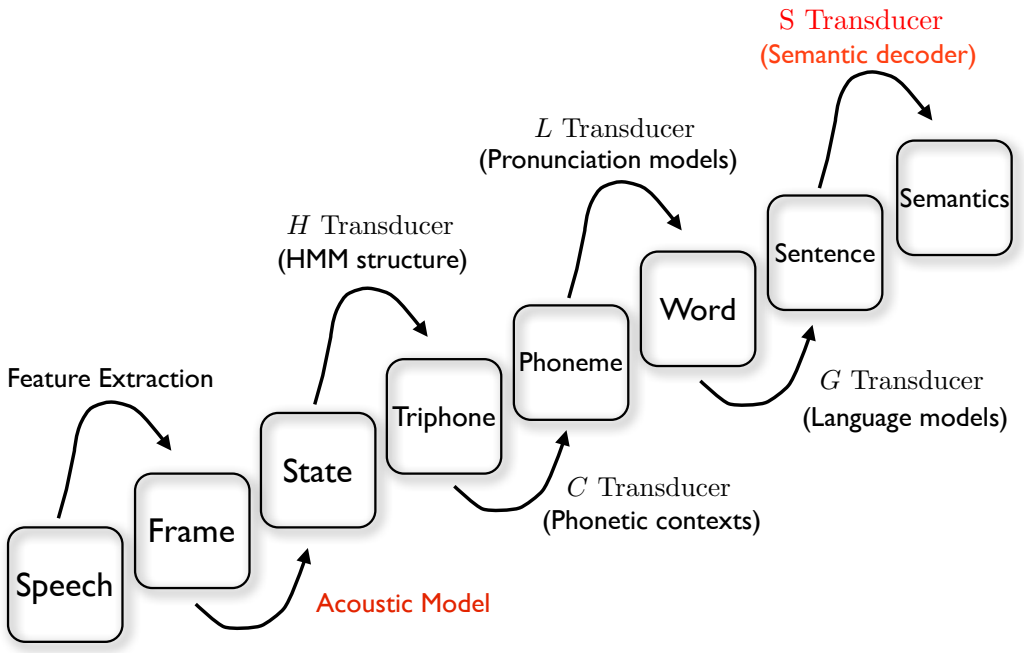

Figure 5: The diagram of speech recognition and understanding in a WFST view

1.4 Motivations and Scientific Goals

The objective of this thesis is to build a conversational speech recognition and understanding system which is robust to various adverse acoustical environments. With the challenges we are facing, the motivations are the following,

- The WERs of word-by-word transcription on a conversational speech task are relatively high. Observing the fact that the semantic information of a conversational speech utterance is usually embedded in the set of keywords, the problem of keyword spotting becomes crucial in the conversational speech scenarios.

- Adverse acoustical environments degrades the ASR system performance substantially. Recently, DNN based acoustic modeling has shown great success on LVCSR tasks. This opens new possibilities for further improving the environmental robustness in recognizing the conversational speech.

- Due to the frequently spoken fillers, disfluencies and functional words in the conversational speech, a robust semantic decoder is needed to work seamlessly with the

ASR outputs.

With respect to the motivations listed above, we have the following scientific goals to accomplish,

I propose a model training methodology for keyword spotting for conversational speech understanding

II propose DNN based acoustic models that are robust to additive noise, channel distortions and interference of competing talkers

III propose a robust WFST based semantic decoder seamlessly coupling with ASR.

IV propose a framework to integrate the proposed methods above to a final system

1.5 Dissertation Outline

We will focus on the acoustic modeling part in Chapter 2-4 and the semantic decoder part in Chapter 5. The integration of the two parts will be described in Chapter 6.

In Chapter 2, we propose non-uniform MCE approach and present how it can be implemented efficiently in the WFST framework. We show the proposed framework can achieve consistent and significant spotting performance gains on two challenging large-scale spontaneous conversational telephone speech (CTS) datasets in two different languages (English and Mandarin). Goal I (c.f., Section 1.4) is accomplished in this chapter.

In Chapter 3, we propose recurrent DNN-HMM systems for robust ASR. A new back-propagation through time (BPTT) algorithm is introduced to make the minibatch stochastic gradient descent (SGD) more efficient and effective. With the proposed system, the state-of-the-art performances are achieved on the 2nd CHiME challenge (track 2) and Aurora-4 tasks without front-end preprocessing, speaker adaptive training or multiple decoding passes.

In Chapter 4, we study the problem of single-channel mixed speech recognition using DNNs. We investigate several different multi-style training setups and introduce a WFST-based two-talker decoder to work with the trained DNNs. The best setup of the proposed systems achieves an overall WER of 18.8% which improves upon the previous state-of-the-art system by 2.8% absolute. Goal II is accomplished in Chapter 3 and this chapter.

In Chapter 5, we propose latent semantic rational kernels (LSRKs) framework for topic spotting on conversational speech. We also present how to generalize the LSRKs using tf-idf weighting, latent semantic analysis, WordNet and probabilistic topic models. With the proposed LSRK, the significant and consistent topic spotting performance gains are achieved on the Switchboard and AT&T HMIHY0300 initial collection. Goal III is accomplished in this chapter.

In Chapter 6, we propose non-uniform sequential DT of DNNs with LSRKs which directly links the information of the semantic decoder to the objective function of the DT. The experimental results on the subset of Switchboard demonstrate the proposed method can lead the acoustic modeling to a more robust system with respect to the semantic decoder. Goal IV is accomplished in this chapter.

In Chapter 7, we conclude the thesis by listing the contributions and future perspectives.

CHAPTER 2

NON-UNIFORM MINIMUM CLASSIFICATION ERROR FOR KEYWORD SPOTTING

2.1 Introduction

The complexity of a general automatic speech recognition (ASR) task is characterized along two dimensions: the size of the vocabulary and the speaking style [33]. One usually can expect higher than 90% word accuracy, which is calculated using the Levenshtein distance between the label and the fully recognized transcriptions, on a large vocabulary task with speech that is either read or rendered in a dictation speaking style, *e.g.*, the WSJ task. This accuracy, however, would dramatically decrease for a spontaneous conversational task, *e.g.*, the Switchboard, where the speaking style is much less constrained than a reading monologue. It is also generally true that a word-by-word transcription of a spontaneous speech signal may not be necessary in many scenarios. In automation which requires some limited semantic understanding of a naturally spoken utterance, it is generally assumed the relevant semantic notions are embedded in the set of keywords [34]. (The situation is similar in surveillance.) One good example is AT&T's How-May-I-Help-You (HMIHY) [35] system, in which the concept of *salient* words is proposed to evaluate the various word significance using the mutual information between each word and the call-type for call-routing services.

Therefore, keyword spotting techniques for the ASR problem become crucial in spontaneous speech scenarios. Keywords spotting, a key technological component in the 1970s, was primarily based on template matching using dynamic time warping (DTW) [36]. Recent systems are mostly based on hidden Markov models (HMMs). In spite of several engineering successes, such as AT&T's VRCP (voice recognition call processing) [37], a formal formulation of the keyword spotting problem in the spirit of hypothesis testing remains elusive. In [38], the system employs N whole-word HMMs to represent N keywords,

and an additional model, *i.e.*, the filler or garbage model, to represent those non-keyword speech signals; Keyword spotting is then carried out as an (N+1)-word recognition, possibly followed by a hypothesis testing step using likelihood ratios. To better capture the characteristics of non-keywords, more filler models and the related structures may be introduced into the grammar network, which treats the keyword spotting problem as an (N+M)-word recognition problem [39]. Obviously, the performance of these whole-word model systems are often hampered by the issue of insufficient training data, thus giving rise to the modern use of sub-word based models. More reliable model estimation may be achieved by constructing keyword models as concatenations of phonetic HMMs. More recently, benefited from large vocabulary continuous speech recognition (LVCSR) techniques, a two-stage approach [40] is often shown to deliver good word-spotting results. In the first stage, the approach uses an LVCSR decoder to produce a set of hypothesized transcriptions, from which the presence of keywords are detected and verified in the second stage. The key issue in this approach is that the two stages are isolated and most likely designed under different criteria. LVCSR systems are trained to minimize the word error rate (WER) in general, without placing any emphasis on those keywords.

Discriminative training (DT) [24] [25] [26] is a general technique to boost the recognition accuracy of an LVCSR system; it optimizes the model parameters to "minimize" recognition errors. If we envision word-spotting as a recognition task in which only the recognition accuracy of some words (*i.e.*, those keywords) out of all possible words in the vocabulary needs to be maximized, an adaptation of the fundamental principle of error minimization in DT may present a new paradigm for performance enhancement in word-spotting. Such an adaptation would call for the introduction of non-uniform error cost embedded in discriminative training and our prior work [41] [42] [43] on DT using non-uniform criteria point to a perfect candidate. Thus, in [44], we generalized keyword spotting as a non-uniform error ASR problem, successfully applied our DT algorithms using non-uniform criteria to it and proposed the method of *non-uniform MCE*. The main

idea is to adapt the fundamental MCE criteria in a cost-sensitive way which leads optimizations to place emphasis on keywords. Specifically, MCE [25] was first recast with the embedding of the error cost functions and then implemented efficiently in the weighted finite-state transducer (WFST) framework to fit a keyword spotting task. It is shown that even with a quite simple error cost function, we can achieve considerable and consistent spotting performance gains on a spontaneous conversational speech task.

To further boost the spotting performance and tackle the potential issue of over-training in non-uniform MCE, we presented a complete framework of DT using non-uniform criteria for keyword spotting, namely *the adaptive boosted non-uniform MCE*, in [45]. To be specific, we first make two improvements to the fundamental MCE optimization procedure as in Boosted MMI [27], *i.e.*, canceling any shared part of the numerator and denominator statistics on each frame and replacing I-smoothing to ML estimate with the one to the previous iteration's value. With the two improvements, our MCE implementation in the WFST framework as in [44] can obtain comparable word accuracy gains with both Boosted MMI and MPE [26]. On top of this boosted MCE and motivated by AdaBoost [46], we further introduce an adaptive scheme to embed error cost functions, namely the adaptive adjustment of the error cost function depending on whether the current frame is classified correctly or not, together with the model combinations during the decoding procedure. With the proposed framework, we further improved the spotting performance of non-uniform MCE and achieved significant and consistent figure of merit (FOM) gains over both ML and discriminatively trained systems.

In this chapter, we integrate our two earlier works [44] [45] on DT using non-uniform criteria for keyword spotting to form a more complete and thorough derivation of the proposed framework and present more details regarding its implementation. And we comprehensively validate it on two challenging large-scale spontaneous conversational telephone speech (CTS) tasks in different languages (English and Mandarin) with more experimental results and analysis.

The remainder of this chapter is organized as follows. We generalize our previous works on DT using non-uniform criteria and formulate non-uniform MCE in Section 2.2. Then the implementation of non-uniform MCE in the WFST framework is described in Section 2.3. The complete set of algorithms and implementation details of the adaptive boosted non-uniform MCE are given in Section 2.4. We report experimental results with the analysis and discussions in Section 2.5 and draw our conclusions in Section 2.6.

2.2 Non-uniform MCE Formulations

The problem of keyword spotting can be formulated in various ways. It can be cast as a detection problem, entailing a formulation that follows the Neyman-Pearson Lemma. A keyword is a target event to be detected in a sequence of possibly noisy observations. It can also be formulated as an (N+1)-word recognition problem (or N+M if a more detailed representation of non-keyword events is desirable), as explained earlier. The focus of this chapter is about yet another and perhaps the most versatile formulation, based on the concept that in LVCSR recognition decisions on keywords bear substantially more significant consequences than non-keywords. As alluded to in Section 2.1, this formulation is possible because of the recent advances in non-uniform error cost modeling and decoding [41] [42] [43] [44] [45]. In this section, we will review the general MCE DT method, generalize our previous work on DT using non-uniform error cost criteria for keyword spotting, and provide algorithmic details on how this formulation can be practically implemented with good results.

2.2.1 Discriminative Training Based on MCE

The general MCE training is a DT method for pattern recognition with the aim of direct minimization of the *empirical error rate*. In the speech recognition scenario, let X_r, $r = 1, \cdots, R$, be the utterances in the training set, W_r be the transcription word label for X_r and W be the set of selected hypothesis events. The discriminant function for a hypothesis W

is defined as,

$$g(X_r, W; \theta) = \log\left[P(X_r|W;\theta)^{\kappa} P(W)\right], \tag{14}$$

where $P(X_r|W;\theta)$ and $P(W)$ denote the acoustic and language models respectively, and κ is the corresponding acoustic model scaling factors. Thus the misclassification measure takes the following form,

$$\begin{aligned} d(X_r;\theta) &= -g(X_r, W_r;\theta) \\ &\quad + \log\left[\frac{1}{|W|}\sum_{W\neq W_r}\exp[g(X_r, W;\theta)]\eta\right]^{\frac{1}{\eta}}. \end{aligned} \tag{15}$$

Finally, with proper smoothing using the sigmoid function, the objective function is formulated as,

$$\mathcal{L}(\theta) = \sum_{r=1}^{R} \ell(d(X_r;\theta)), \tag{16}$$

where $\ell(\cdot)$ is the sigmoid function,

$$\ell(d) = \frac{1}{1+\exp(-\alpha d+\beta)}. \tag{17}$$

As can be seen from (16), the objective function forms a smoothed approximation of *the empirical errors* with respect to the training set. The model parameters can be optimized iteratively to minimize the objective function via the generalized probabilistic descent (GPD) algorithm as in the original MCE work or the gradient descent (GD) and the extended Baum-Welch (EBW) after rewriting the objective function according to [47].

2.2.2 General DT Using Non-uniform Criteria

The main idea of DT using non-uniform criteria is motivated by the fact that some recognition units may carry more significance than others and the objective of training should result in a minimized error cost rather than a minimized error rate, which does not differentiate among various errors. Consistent with the original Bayes decision theory, the minimum classification error cost (MCEC) methodology was proposed for pattern recognition in [41] and the authors also demonstrated several application scenarios of the extended

framework. Here, we follow the same spirit of MCE and MCEC and extend it to sequential modeling and decoding, which is necessary because keywords are likely to be embedded in natural continuously-spoken sentences, consisting of a multiplicity of word (or subword) units. Our discussions below thus focus on the adaptation of MCEC in an LVCSR task because some approximations and simplifications become necessary when implementation issues are taken into account.

In the original Bayes decision theory, the risk associated with a decision of recognizing a pattern X as a C_i event is defined as,

$$R(C_i|X) = \sum_{j \in I_C} \epsilon_{ij} P(C_j|X) \approx \sum_{j \in I_C} \epsilon_{ij} \mathcal{H}_j(X;\theta). \tag{18}$$

In (18), $P(C_j|X)$ is the true aposteriori probability for class C_j, which is ideal and modeled by $\mathcal{H}_j(X;\theta)$ with parameter θ. The error assignment ϵ_{ij} defines the cost of recognizing a j^{th} class event as an i^{th} class event. For the non-uniform error cost formulation of [41], this error assignment needs not be a 0-1 function (*i.e.*, an error count) and is preserved in the synthesis of a proper empirical risk function for optimization. Note that in this original formulation, the observed X is a realized whole pattern corresponding to an event class, albeit unknown. The recognizer function $C(X)$ for a given X (upon which a decision is to be rendered) is implemented as,

$$C(X;\theta) = \arg\min_{i \in I_C} R(C_i|X) = \arg\min_{i \in I_C} \sum_{j \in I_C} \epsilon_{ij} \mathcal{H}_j(X;\theta). \tag{19}$$

For convenience, we use the notation $R(C_i|X) = \sum_{j \in I_C} \epsilon_{ij} \mathcal{H}_j(X;\theta) = \boldsymbol{\epsilon}_i^T \mathbf{h}(X;\theta)$, where $\boldsymbol{\epsilon}_i$ and $\mathbf{h}(X;\theta)$ are column vectors, $\boldsymbol{\epsilon}_i = [\epsilon_{i1}, \epsilon_{i2}, \cdots, \epsilon_{iI_C}]^T$, $\mathbf{h}(X) = [\mathcal{H}_1(X;\theta), \mathcal{H}_2(X;\theta), \cdots, \mathcal{H}_{I_C}(X;\theta)]^T$. Typically, $\epsilon_{jj} = 0$, $\epsilon_{ij} > 0$ for $i \neq j$, and $E_{X \in C_j}[\boldsymbol{\epsilon}_i^T \mathbf{h}(X)] \geq E_{X \in C_j}[\boldsymbol{\epsilon}_j^T \mathbf{h}(X)]$ for random observations that belong to j^{th} class. This condition can be considered a basic operational requirement because if it is not true, the class C_j should be eliminated or joined with some other class. The empirical non-uniform cost to be minimized based on a training

set $\Omega = \{X\}$ can thus be written as,

$$\mathcal{R}(\Omega;\theta) = \sum_{X\in\Omega}\left[\min_{i\in I_C} R(C_i|X)\right] \approx \sum_{X\in\Omega}\min_{i\in I_C}[\boldsymbol{\epsilon}_i^T\mathbf{h}(X)], \tag{20}$$

and

$$\theta^* = \arg\min_{\theta}\mathcal{R}(\Omega;\theta), \tag{21}$$

is the result of error cost minimization, to be used as the system model set for implementing the recognizer of (19). The class label information in (20) can be made explicit during training by rewriting (20) as

$$\mathcal{R}(\Omega;\theta) = \sum_{j\in I_C}\sum_{X\in C_j}\min_{i\in I_C}[\boldsymbol{\epsilon}_i^T\mathbf{h}(X)]. \tag{22}$$

Note that $E[\boldsymbol{\epsilon}_j^T\mathbf{h}(X)] = B_j = \min_i E[\boldsymbol{\epsilon}_i^T\mathbf{h}(\mathbf{X})]$ for X in C_j where B_j is the Bayes (minimum) risk for class C_j and $B = E[B_j]$ is the Bayes (minimum) risk. Bayes risk is the theoretical limit of the achievable minimum,

$$\sum_{X\in C_j}\min_{i\in I_C}[\boldsymbol{\epsilon}_i^T\mathbf{h}(X)] - B_j = \sum_{X\in C_j}\min_{i\in I_C}[\boldsymbol{\epsilon}_i^T\mathbf{h}(X) - B_j]. \tag{23}$$

It means for a given j^{th} class token, X, if $\min_i[\boldsymbol{\epsilon}_i^T\mathbf{h}(X)] = \boldsymbol{\epsilon}_j^T\mathbf{h}(X)$, there is no need to adjust the parameters in θ as the solution will readily lead to the Bayes risk (the minimum cost incurred in rendering the correct decision only contributes to a possible estimate of the Bayes risk). Therefore, the parameter adjustment can focus on those tokens that do not lead to the decision C_j and the system optimization objective can be defined based on (for class C_j tokens),

$$\sum_{X\in C_j}\mathcal{L}(X;\theta) = \sum_{X\in C_j}\ell\left(\boldsymbol{\epsilon}_j^T\mathbf{h}(X) - \min_i[\boldsymbol{\epsilon}_i^T\mathbf{h}(X)]\right), \tag{24}$$

where $\ell(\cdot)$ is a hinge loss function. The objective of (24) can be interpreted as follows. For a class C_j token, if the cost $\boldsymbol{\epsilon}_i^T\mathbf{h}(X)$ is not the minimum among all $i \in I_C$, the hinge loss will be positive, requiring adjustment of parameter values to minimize the loss.

When applied to continuous speech recognition, suppose now the r^{th} utterance X_r in the training set is segmented in to S_r tokens, each corresponding to a unitary decision (in

accordance with the designation of the model set $\theta = \{\mathcal{H}_i(X;\theta)\}$), the general formulation of DT using non-uniform criteria is now given by,

$$\mathcal{L}(\theta) = \sum_{r=1}^{R}\sum_{s=1}^{S_r}\sum_{j\in I_C} \ell\left(\boldsymbol{\epsilon}_j^T \mathbf{h}(X_r^s) - \min_i[\boldsymbol{\epsilon}_i^T \mathbf{h}(X_r^s)]\right) \mathbb{1}\{X_r^s \in C_j\}. \tag{25}$$

The knowledge of label has been embedded in the indicator function as well as in the argument for the hinge loss, under the assumption that the optimal decoded segmentation is already in place. $\mathbf{h}(X_r^s)$ is the column vector where each entry corresponds to $\mathcal{H}_i(X_r^s;\theta)$, the discriminant function for the class C_i; it assumes the role of the posterior probability of C_i in Bayes decision theory. And $\boldsymbol{\epsilon} = \left[\boldsymbol{\epsilon}_i^T\right]_{i\in I_C}$ is the error cost matrix. Note that the segmentation may be defined in different granularities; that is, the segmented tokens X_r^s may correspond to phone(me)s, words, or phrases, depending on the pre-specified construct of the class model set with a set of non-uniform error cost assignments at the corresponding unit class level. It is, however, also worth noting that sometimes ϵ_{ij} needs to be defined in a cross-level fashion, referred to as the inter-level case in [41], where the level that we measure the system performance on may be different from the level of the models we use. The identity of C_i also implies the level of modeling and decoding units.

The evaluation of the term $\min_i[\boldsymbol{\epsilon}_i^T \mathbf{h}(X_r^s)]$ in (25) involves finding the class label with minimum risk over all hypothesis class. In LVCSR, this evaluation is compounded by the vast hypothesis space, making the determination and representation of the word lattices or the phone networks prohibitively difficult. This entanglement with the decoded segmentation makes the optimal implementation unwieldy if not impossible (One way is first converting the word or phone lattice to confusion network [48]. Then minimum Bayes' risk decoding [49] is employed base on the network beforehand to get the segmentation).

As suggested in [41], one alternative, which is asymptotically equivalent in the context of an empirical estimate of the risk, is the post-decision risk, as investigated in our previous

work [43], defined as

$$\mathcal{L}(\theta) = \sum_{r=1}^{R} \sum_{s=1}^{S_r} \sum_{i \in I_C} \sum_{j \in I_C} \epsilon_{ij} \cdot \ell \left(-\mathcal{H}_j(X_r^s) + \max_i [\mathcal{H}_i(X_r^s)] \right) \mathbb{1} \left\{ i = \arg \max_{i' \neq j, i' \in I_C} \mathcal{H}_{i'}(X_r^s; \theta) \right\} \mathbb{1} \left\{ X_r^s \in C_j \right\}. \tag{26}$$

The argument for the hinge loss is now of the MCE style, in which the evaluated posteriors (model) are being compared to render a decision of i, which incurs a loss of ϵ_{ij}. This simplification, when viewed in the computed hypothesis space, will thus lead to a reasonable construct of the empirical risk that reasonably approximates that of (18). The post-decision risk of (26), assuming that the class events are defined at the phone level, means only the error cost ϵ_{ij} corresponding to the most probable phonetic hypothesis over a certain time (segmentation) interval is considered.

In the keyword spotting scenario, however, it is desirable to broaden the hypothesis space to avoid premature occurrences of the two types of errors, namely false alarm and miss. To this regard, if the class events are still defined at the phone level, various hypothesis phone alignments will exacerbate the incorporation of non-uniform error assignments during decoding as discussed earlier. Furthermore, to take advantage of the FST implementation (see section 2.3), we decide that the segment based cost assignment is carried out on a frame basis. In the following subsection, we will make the necessary adaptation which allows the sequential introduction of the non-uniform error cost on the frame level to formulate the non-uniform MCE for keyword spotting.

2.2.3 Non-uniform MCE for Keyword Spotting

To formulate the non-uniform MCE for keyword spotting, let us first examine the error cost function. Let $\mathcal{V} = \{w_i\}_{i=1}^{V}$ be the vocabulary consisting of V words and ϵ_{ij} the error cost function which specifies the error costs in identifying a word w_j word as the hypothesized w_i in the vocabulary. The performance metric of a conventional LVCSR system is defined by the WER, which when cast in Bayes' optimal decision theory assumes 0-1 loss function. Without loss of generality, let $\mathcal{K} = \{w_i\}_{i=1}^{K}$ be the set of chosen keywords. The task of

keyword spotting is to detect these keywords, which may be embedded and realized in a sentence, while making no particular identification of other words in the vocabulary. This descriptive task statement can be formulated in the same LVCSR framework with the following error cost matrix (function),

$$\boldsymbol{\epsilon}_{V\times V} = \begin{bmatrix} 0 & \epsilon_{1,2} & \cdots & \epsilon_{1K} & \epsilon_{1,K+1} & \cdots & \epsilon_{1V} \\ \epsilon_{2,1} & 0 & \cdots & \epsilon_{2K} & \epsilon_{2,K+1} & \cdots & \epsilon_{2V} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \epsilon_{K1} & \epsilon_{K2} & \cdots & 0 & \epsilon_{K,K+1} & \cdots & \epsilon_{KV} \\ \epsilon_{K+1,1} & \epsilon_{K+1,2} & \cdots & \epsilon_{K+1,K} & 0 & \cdots & 0 \\ \epsilon_{K+2,1} & \epsilon_{K+2,2} & \cdots & \epsilon_{K+2,K} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \epsilon_{V,1} & \epsilon_{V,2} & \cdots & \epsilon_{V,K} & 0 & \cdots & 0 \end{bmatrix}. \tag{27}$$

The upper left part of the error cost matrix allows the non-uniform assignment of cost when a particular keyword w_j is mistaken as another one, w_i; the lower left and upper right corners represent the cost of the two types of error, namely, missing a keyword (detection miss) and false recognition of a non-keyword as a keyword (false alarm), respectively; The lower right part indicates misrecognition of non-keywords contributes nothing to the performance. Invoking the above error cost matrix in the original Bayes' optimal decision theory leads to a rigorous and formal framework both for discriminative training and decoding for the keyword spotting, aiming at the direct *error cost* minimization. Now the issue is to find ways to assign the error cost taking into account the implementation details.

The original formulation of non-uniform error cost minimization [41] is a general framework for pattern recognition when class-dependent errors (a function of both the ground truth and the system decision) are desirable. When extended to continuous speech recognition, it involves the empirical cost of (25), which implies the availability of segmented data. In this chapter, we propose an alternative introduction and interpretation of the non-uniform error cost in the context of efficient implementation. This is accomplished first by taking advantage of the conventional segmental HMM structure which implies that

each word/unit hypothesis is associated with a likelihood or posterior. This allows sequential introduction of the non-uniform error cost on a frame basis. In the following, we redraw the implementation of non-uniform error cost minimization based on frame-based modification. First for certain frame t, we define the frame discriminant function as,

$$\mathcal{H}_i(X_r^t, W; \theta) = P(s_i|X_r, W) \log \mathcal{N}_i(X_r^t, \theta), \tag{28}$$

where i is the state index associated with certain Gaussian $\mathcal{N}_i(\cdot)$, and $P(s_i|X_r, W)$ is the state posteriors (occupancy probability) given observations X_r and hypothesis transcription W. In keyword spotting, we need to broaden the hypothesis space to detect premature occurrences of the two types of errors, which means only considering the most probable hypothesis as in (26) is not enough while we only need to care about whether the keywords occurs in the label word sequence W_r or the hypothesis $W \neq W_r$ at t^{th} training frame. Therefore, with the frame discriminant function as in (28) and the error cost matrix as in (27), (26) now can be modified as,

$$\mathcal{L}(\theta) = \sum_{r=1}^{R} \sum_{t=1}^{T_r} \sum_{j \in I_C} \ell \left(-\mathcal{H}_j(X_r^t, W_r) + \sum_{i, W \neq W_r} \mathcal{H}_i(X_r^t, W) \right)$$
$$\epsilon_{ij} \cdot \mathbb{1} \left\{ (i \in \mathcal{K}, W \neq W_r) \vee (j \in \mathcal{K}, W_r) \right\} \mathbb{1} \left\{ X_r^s \in C_j \right\}. \tag{29}$$

The formulated objective function above only considers the incurred risk when the keywords occurs in label or hypothesis word sequence, to allow more general error cost assignment, we define $\epsilon_r(t)$ as the error cost function, which specifies the values of error cost over time (frame) through the r^{th} utterance, the objective function of non-uniform MCE for keyword spotting can be written as,

$$\mathcal{L}(\theta) = \sum_{r=1}^{R} \sum_{t=1}^{T_r} \epsilon_r(t) \ell[-\mathcal{H}_j(X_r^t, W_r) + \sum_{i, W \neq W_r} \mathcal{H}_i(X_r^t, W)]. \tag{30}$$

To gain an insight into the objective function of non-uniform MCE in (30), we write down the gradient of this accumulated risk for Gaussian mixture HMMs (for convenience, we let $\gamma_{jm}^{W_r}(t)$ and $\gamma_{jm}^{W \neq W_r}(t)$ be state posteriors of the Gaussian at certain frame t for the label and

the hypothesized transcriptions respectively),

$$\nabla\mathcal{L}(\theta) = \sum_{r=1}^{R}\sum_{t=1}^{T_r} \ell(d(X_r^t;\theta))[1-\ell(d(X_r^t;\theta))] \\ \epsilon_r(t)\left(-\gamma_{jm}^{W_r}(t) + \gamma_{jm}^{W\neq W_r}(t)\right)\frac{\partial \log \mathcal{N}_{jm}(X_r^t,\theta)}{\partial\theta}, \tag{31}$$

where $\mathcal{N}_{jm}(X_r^t,\theta)$ is the corresponding Gaussian of certain model j and mixture m. If we compare this with the gradients of regular MCE objective function (for simplification, we let $\eta = 1$ and ignore the factor $1/|W|$ in (15)),

$$\nabla\mathcal{L}_{\text{MCE}}(\theta) = \sum_{r=1}^{R}\sum_{t=1}^{T_r} \ell(d(X_r;\theta))[1-\ell(d(X_r;\theta))] \\ \left(-\gamma_{jm}^{W_r}(t) + \gamma_{jm}^{W\neq W_r}(t)\right)\frac{\partial \log \mathcal{N}_{jm}(X_r^t,\theta)}{\partial\theta}. \tag{32}$$

As can be seen, one difference is that for the non-uniform MCE, the value of $\ell(d(X_r^t;\theta))[1-\ell(d(X_r^t;\theta))]$ changes over frames thus needs to be evaluated frame by frame, while for the regular MCE, it is fixed value through one utterance. The other difference is non-uniform error cost function $\epsilon_r(t)$ is imposed on each individual frame. To make the implementation of non-uniform MCE can be easily recast from the regular MCE, we bring the first difference into agreement and make an approximation for the gradients of the non-uniform MCE objective function as,

$$\nabla\mathcal{L}(\theta) \approx \sum_{r=1}^{R}\sum_{t=1}^{T_r} \gamma\ell(d(X_r;\theta))[1-\ell(d(X_r;\theta))] \\ \epsilon_r(t)\left(-\gamma_{jm}^{W_r}(t) + \gamma_{jm}^{W\neq W_r}(t)\right)\frac{\partial \log \mathcal{N}_{jm}(X_r^t,\theta)}{\partial\theta}. \tag{33}$$

The value of the non-uniform error cost function $\epsilon_r(t)$ at t^{th} frame can be absorbed into the corresponding occupancy probabilities as,

$$\nabla\mathcal{L}(\theta) \approx \sum_{r=1}^{R}\sum_{t=1}^{T_r} \gamma\ell(d(X_r;\theta))[1-\ell(d(X_r;\theta))] \\ \left[-\epsilon_r(t)\cdot\gamma_{jm}^{W_r}(t) + \epsilon_r(t)\cdot\gamma_{jm}^{W\neq W_r}(t)\right]\frac{\partial \log \mathcal{N}_{jm}(X_r^t,\theta)}{\partial\theta}. \tag{34}$$

This implies after the embedding of the error cost function the number of times that the training procedure of non-uniform MCE sees the training sample at t^{th} frame is scaled by

the value of $\epsilon_r(t)$. From another perspective, suppose the training sample X_r^t is drawn from a certain distribution $\mathcal{D}(X_r^t)$, employing non-uniform MCE with the error cost function embedded on the original training set is equivalent to employing the regular MCE on an artificial *resampled* training set which observes the distribution,

$$\hat{\mathcal{D}}(X_r^t) = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\epsilon_r(t)\mathcal{D}(X_r^t)}{E_{\epsilon_r(t),X_r^t\sim\mathcal{D}(X_r^t)}[\epsilon_r(t)]}. \tag{35}$$

It can be shown in [50] that any *error rate* minimizing classifier on the *resampled* distribution $\hat{\mathcal{D}}$, whereby the probability of a certain sample is proportionate to its error cost, will accomplish expected *error cost* minimization on the original distribution $\mathcal{D}$ under the assumption that the training samples are drawn independently from the corresponding distributions.

For a keyword spotting task, as $\epsilon_r(t)$ is a function over the speech frames, it is straightforward to design it in a way that all frames labeled as keywords should be assigned a higher value. At the same time, as mentioned earlier, those frames with high possibility recognized as keyword hypothesis should also be emphasized to prevent the false alarms. This can be efficiently done via searching keywords in the corresponding decoded lattice and recording the start and end frames. One simple example of the error cost function design used for the keyword spotting can be as follows,

$$\epsilon_r(t) = \begin{cases} 2 & t \in \{t|W_r(t) \in \mathcal{K}\} \\ 2 & t \in \{t|W(t) \in \mathcal{K}\} \\ 1 & \text{otherwise} \end{cases}. \tag{36}$$

One can also design the error cost function in an asymmetrical way, *i.e.*, using different values when keywords occurs in reference and hypothesis respectively, to achieve a desirable compromise between the miss-detection and false-alarm rate. Considering that the keyword's contextual frames may also need to be given additional emphasis, we can accordingly enlarge the error cost for the frames near the boundaries between keyword and non-keyword frames.

2.3 Non-uniform MCE Implementation in the WFST Framework

In this section, we describe how non-uniform MCE can be efficiently implemented in the WFST framework. The WFST decoding graph for an ASR problem [23] can be written as,

$$HCLG = \min(\det(H \circ C \circ L \circ G)), \tag{37}$$

where H, C, L and G are the HMM structure, phonetic context-dependency, lexicon and grammar FST respectively. For a more intuitive presentation, we use the terms "costs" or "scores" to refer to the weights of a WFST used in the ASR. In $HCLG$, the input labels are the identifiers (indices) of context-dependent HMM states, and the output labels represent words. The special symbol ϵ which denotes the empty labels may appear in both the input and output labels.

If we want to decode an utterance of T frames, *i.e.*, find the most likely word sequence and its corresponding state-level alignment, a WFST interpretation of the decoding problem [51] is as follows: Consider a $(T+1)$-state WFSA U in which both input and output labels of its arcs (transitions) are the identifiers of context-dependent HMM states and the cost of each arc outgoing from t^{th} state to $(t+1)^{\text{th}}$ state is the negated (perhaps scaled) log-likelihood of t^{th} frame observation given the probability density function (pdf) corresponding to its input or output label indexed HMM state. Then the full search graph for this utterance is expressed as,

$$S = U \circ HCLG. \tag{38}$$

The decoding is thus equivalent to finding the best path with shortest distance (cost) through S. The input label sequence for this best path represents the state-level alignment, and the output label sequence is the corresponding sentence. Since in practice S is very large, a decoded *lattice*, which forms a compact representation of the hypothesis space for this utterance, is usually a beam-pruned subgraph of S,

$$P = \text{prune}(S, \alpha), \tag{39}$$

where α is the pruning beam width. But P is not exactly the hypothesis space we need in non-uniform MCE. As in (16), the competing hypothesis for MCE training has to exclude the reference word transcription W_r. Naively removing the arcs corresponding to the reference words will hurt the topological structures of the decoded lattice. Although this issue can be circumvented via subtracting the corresponding reference's statistics from the nominator and denominator of posterior in [47], we can come up with a more ingenious solution that takes advantage of WFST's *difference* operation to exclude the reference. Since only strings that are in the first operand WFST but not in second operand are retained in the resultant WFST after the difference operation, we can first compile a linear WFST which only accepts the reference word string then subtracts it from the decoded lattice.

However, the input symbols of the decoded lattice P are the identifiers of the HMM states, which is not a valid operand for the WFST difference operation we want. In [51], a compact version of the decoded lattice is proposed whereby the decoded lattice is a WFSA with identical input and output symbols being words, while the acoustic, language score and the state alignment strings are all encoded in to the weight using a special semiring. Specifically, let (c, s) be a pair of the cost c (including both acoustic and language cost) and a state symbol sequence s,

$$(c, s) \otimes (c', s') = (c + c', (s, s')), \tag{40}$$

$$(c, s) \oplus (c', s') = \begin{cases} (c, s) & \text{if } c < c' \\ (c', s') & \text{if } c > c' \\ (c, s) & \text{if } c = c',\ \text{len}(s) < \text{len}(s') \\ (c', s') & \text{if } c = c',\ \text{len}(s) > \text{len}(s') \end{cases}, \tag{41}$$

where (s, s') is the concatenation of s and s', if both the costs and the length of the state strings are identical, the $\oplus$ operator will return the pair whose string appears first in the dictionary order. Denote by L the compact version of the decoded lattice P, the generation of L can be summarized as follows. First P is inverted to P^{-1}, and then P^{-1} is encoded into a WFSA E with the same number of states and arcs. Both input and output symbols of E

are the input symbols (words) on the corresponding arcs of P^{-1} and the weights contains both cost and the output symbols (HMM state identifiers) on the corresponding arcs of P^{-1}. Finally the compact version of P is given by,

$$L = \det(\text{rmeps}(E)), \tag{42}$$

where rmeps denotes the ϵ removal operation, and the determinization operation det is conducted based on the semiring defined in (40) and (41). It can be seen from the definition of $\otimes$ only the best state alignment with lowest cost for each word sequence is kept in L after the determinization operation. With this compact version of the decoded lattice, we can conduct the WFST difference and generate the lattice for MCE training. Denote by $L_r(W)$ the compact version of the decoded lattice for r^{th} utterance and let $\text{WFST}(W_r)$ be the compiled WFST for r^{th} utterance's label transcription, the lattice for the MCE training is given by,

$$L_r^{\text{MCE}} = L_r(W) - \text{WFST}(W_r). \tag{43}$$

With L_r^{MCE} available, $\gamma_{jm}^{W \neq W_r}(t)$ can be directly evaluated using forward-backwards on it. Then for the embedding of the non-uniform error cost, we only need a vector with its length equal to the frame number of the utterance, and search for the keywords in both label alignments and L_r^{MCE}, recording the time intervals where the keywords occur respectively and setting the corresponding value of the vector, then use this vector to scale the posteriors when updating the model parameters. Therefore, in the WFST framework, the non-uniform MCE can be implemented efficiently without significant additional overheads compared to the regular MCE.

2.4 Adaptive Boosted Non-uniform MCE Framework for Keyword Spotting

In this section, to further boost the spotting performance and tackle the potential issue of over-training in the non-uniform MCE, we present a complete framework of DT using non-uniform criteria for keyword spotting, *adaptive boosted non-uniform MCE*. We first make

two improvements to the fundamental MCE optimization procedure and then introduce an adaptive scheme to embed error cost functions together with the model combinations during the decoding procedure.

2.4.1 Improvements to MCE updates

In the original MCE work [25], the model parameters were optimized via generalized probabilistic descent (GPD) while extended Baum-welch (EBW) [52], a.k.a., conditional expectation-maximization (CEM), appears to work better in LVCSR since determining the step size for GPD is still not trivial. Although there have been several works on this, we use EBW to do the parameter updates. Furthermore, we make two improvements to it as in Boosted MMI: The first is we cancel any shared part of the numerator and denominator posteriors (occupancy probabilities in reference and hypothesis) on each frame,

$$\gamma_{jm}^{W_r}(t) = \gamma_{jm}^{W_r}(t) - \min(\gamma_{jm}^{W_r}(t), \gamma_{jm}^{W \neq W_r}(t)). \tag{44}$$

$$\gamma_{jm}^{W \neq W_r}(t) = \gamma_{jm}^{W \neq W_r}(t) - \min(\gamma_{jm}^{W_r}(t), \gamma_{jm}^{W \neq W_r}(t)). \tag{45}$$

Note that with the canceling the accumulated statistics remains unchanged, while it changes the Gaussian specific learning rate D_{jm} in EBW updates; After canceling the shared part, the numerator statistics can not be directly used in the ML estimate for I-smoothing, another modification is thus we do I-smoothing to the previous iteration rather that ML estimates. The rule for calculating D_{jm} is simply changed to be

$$D_{jm} = \max(\tau + E\gamma_{jm}^{den}, 2D_{jm}^{\min}), \tag{46}$$

where τ is I-smooth factor, $D_{jm}^{\min}$ is the smallest value that leads the covariance matrix being positive definite. These two modifications were reported to boost the word accuracy considerably in [27], and we will show in the Section 2.5 with these two improvements our fundamental MCE implementation in the WFST framework can achieve comparable performance with both Boosted MMI and MPE.

2.4.2 Adaptive Error Cost Function and Model Combination

On top of the boosted MCE above, we can simply adapt it to non-uniform MCE with the embedding of the error cost function $\epsilon_r(t)$ as described in Section 2.3. The quite simple error cost function we used in (36) imposes the same error cost on certain training frame during the different optimization iterations, which could lead to severe over-training when we use fairly large error cost. In addition, the error cost function with no normalization over the whole training set can lead to too aggressive learning rate for each EBW updates when the number of frames corresponding to keywords is large. If we examine non-uniform MCE from another perspective, as in (33), it is actually equivalent to employing the regular MCE on a resampled training set in which each frame is *reweighted* according to $\epsilon_r(t)$. Thus, the boosting based techniques can be applied here naturally which typically consist of iteratively learning weak classifiers with respect to a resampled data distribution and combining them to a final strong classifier. And adaptive boosting (AdaBoost) appears to be a perfect candidate since during each iteration it will adjust the error cost (weight) corresponding to each data sample adaptively. After Freund and Schapire proposed AdaBoost for binary classification, they also generalized it for multiclass problems, AdaBoost.M1 and AdaBoost.M2 [53], which can be summarized in Algorithm 1. However, it is still not straightforward to incorporate multiclass AdaBoost to ASR which is a sequential classification problem. Several issues need to be addressed before it can be applied: how we define the class in this problem, in what level (utterance/phoneme/frame) we manipulate the sample distribution and how we combine the models trained from each iteration to a final stronger one. Previously, there are several works on boosting techniques for ASR. In [54] and [55], both utterance level and frame level boosting for ASR were investigated. Boosting phoneme HMMs was proposed in [56], and a new method for model combination, multiple stream decoding, was also presented. Recently, boosting has been applied in discriminative trained system with the re-estimated phonetic decision trees in model combination [57]. Below we describe how we embed the error cost function adaptively in a

Algorithm 1 Multiclass AdaBoost

Input: T training examples $\{(x_t, y_t)\}_{t=1}^{T}$, $x_t \in \mathcal{X}$, with class labels $y \in \{1, ..., C\}$, and weak classifiers $h_k \in \mathcal{H}$

1: **for** $t = 1, ..., T$ **do**
2: $D_1(t) = 1/T$
3: **end for**
4: **for** $k = 1, ..., K$ **do**
5: Train weak classifier h_k using distribution D_k.
6: Calculate the error of h_k: $\varepsilon_k = \sum_{t:h_k(x_t)\neq y_t} D_k(t)$.
7: If $\varepsilon_k > 1/2$, abort.
8: Set $\beta_k = \varepsilon_k/(1 - \varepsilon_k)$.
9: **for** $t = 1, ..., T$ **do**
10: Update distribution:

$$D_{k+1}(t) = \frac{D_k(t)}{Z_k} \cdot \begin{cases} \beta_k, & \text{if } h_k(x_t) = y_t \\ 1, & \text{otherwise} \end{cases}, \tag{47}$$

where Z_k is the normalization factor to guarantee $D_{k+1}(t)$ is a distribution.
11: **end for**
12: **end for**

Output: $H(x) = \arg\max_y \sum_{k=1}^{K} \log \frac{1}{\beta_k} \cdot \mathbb{1}(h_k(x) = y)$

similar way as AdaBoost and explain how iteratively trained models are combined to a final stronger one in our framework.

Firstly, we work on the frame level as our error cost function $\epsilon_r(t)$ imposes cost over frame by frame. And $\epsilon_r(t)$ would not be initialized uniformly as in Line 2 of Algorithm 1. As in non-uniform MCE for keyword spotting, we will use higher value for frames corresponding to keywords as in (36), while different values can be assigned asymmetrically where keyword frames occur in reference and hypothesis to achieve desirable compromise between the detection miss and false alarm rate, one can also accordingly enlarge the error cost for the frames near keyword boundaries. Most of boosting techniques for ASR works on phoneme classification level, we choose frame level as the classification granularity, mainly for two reasons: First, as we impose error cost on the frame level which implies the data distribution is resampled at frame level during boosting iterative training procedure, classification on frames gives us fine-grained and consistent system; Second, this is also

more convenient for model combination stage later on.

Therefore, in our AdaBoost-like system, the class of acoustic frames is represented by the pdf corresponding to HMM state. (*e.g.*, the corresponding GMM for a GMM-HMM system.) So the number of classes is equal to the number of leaf nodes (distinct acoustic states) of the phonetic decision trees, and this number can easily go beyond several thousand for a LVCSR system which makes the original multiclass AdaBoost intractable to be directly applied here. Thus we make several modifications to the original algorithms, in each iteration we calculate the error cost for each individual class, namely we will use class-specific ε_k^y instead of one global ε_k. At each frame, we consider it is a misclassification error if the value of accumulated state posteriors in hypothesis (denominator lattice) whose corresponding pdf's identifiers are different from the reference is beyond 0.5,

$$\sum_{j \neq y_t} \gamma_j^{W \neq W_r}(t) > 0.5, \tag{48}$$

where $\gamma_j^{W \neq W_r}(t) = \sum_m \gamma_{jm}^{W \neq W_r}(t)$. Then the class-specific *empirical error cost* over the whole training set is given by,

$$\varepsilon_k^y = \sum_{t: y_t \in y} \mathbb{1} \left\{ \sum_{j \neq y} \gamma_j^{W \neq W_r}(t) > 0.5 \right\} \cdot \epsilon_r^k(t). \tag{49}$$

Note that $\gamma_j^{W \neq W_r}(t)$ needs to be recollected for each iteration k. With the empirical error cost available, we can evaluate class-specific β_k^y and use it to do the model combination for each class.

For the model combination part, instead of doing ROVER [58], what we do is more like state-locked multiple-stream decoding as in [56] but implement in a more efficient way under the WFST framework because it does not need multiple-pass decoding. As we keep the phonetic decision tree and HMM transition probabilities the same during non-uniform MCE iteration, in our framework, we use unified pdf indexing through training iterations and compile transition probabilities into decoding WFST graph before we decode certain utterances. The model combination occurs in the acoustic score generation stage:

during the decoding, when the acoustic score over certain frame is demanded, instead of generating from only one model, we calculate the acoustic score (log-likelihood) as the log-linear interpolation between models,

$$\log p(x_t|\mathcal{M}^j) = \frac{1}{Z_j} \sum_{k=1}^{K} \log \frac{1}{\beta_k^j} \cdot \log p(x_t|\mathcal{M}_k^j), \tag{50}$$

where Z_j is the normalization factor such that

$$\frac{1}{Z_j} \sum_{k=1}^{K} \log \frac{1}{\beta_k^j} = 1. \tag{51}$$

However, we find the values of $\log \frac{1}{\beta_k^j}$ are too flat over models trained from each iteration. So we change (50) to,

$$\log p(x_t|\mathcal{M}^j) = \sum_{k=1}^{K} \mathbb{1}\left\{k = \arg\min_k \varepsilon_k^j\right\} \cdot \log p(x_t|\mathcal{M}_k^j), \tag{52}$$

typically we just pick the model for each class with minimum class-specific empirical error cost through all training iterations. Finally we summarize our adaptive boosted non-uniform MCE in Algorithm 2.

2.5 Experiments

In this section, we comprehensively validate the proposed framework on two challenging large-scale CTS tasks in English and Mandarin: For English, we use 200+ hours spontaneous CTS task, Switchboard-1 Release 2 (LDC97S62); For Mandarin, we use 150+ hours spontaneous CTS task, HKUST Mandarin Telephone Speech (LDC2005S15). We will show our methods can achieve significant and consistent spotting performance gains on both tasks.

2.5.1 Experiments on Switchboard

For English, we evaluate our methods on Switchboard-1 Release 2 which is a collection of 2438 two-sided telephone conversations among 543 speakers (302 male, 241 female). Each pair of callers is introduced a topic for discussion and there are about 70 topics were

Algorithm 2 Adaptive Boosted Non-uniform MCE

Input: T training examples (acoustic frames) $\{(x_t, y_t)\}_{t=1}^{T}$, $x_t \in \mathcal{X}$, with class labels $y \in \{1, .., j, .., C\}$, initial model $\mathcal{M}_0 \in \mathcal{M}$.

1: **for** $t = 1, ..., T$ **do**

2: Initialize error cost function $\epsilon_r^0(t)$

$$\epsilon_r^0(t) = \begin{cases} K_1 & t \in \{t | W_r(t) \in \text{keywords}\} \\ K_2 & t \in \{t | W(t) \in \text{keywords}\} \\ 1 & \text{otherwise} \end{cases}, \tag{53}$$

3: **end for**

4: **for** $k = 1, ..., K$ **do**

5: **for** $t = 1, ..., T$ **do**

6: Collecting $\gamma_j^{W_r}(t)$ and $\gamma_j^{W \neq W_r}(t)$ for k^{th} iteration using model $\mathcal{M}_{k-1}$.

7: Update Error Cost function:

$$\epsilon_r^k(t) = \frac{\epsilon_r^{k-1}(t)}{Z_{k-1}} \cdot \begin{cases} 1, & \text{if } \sum_{j \neq y_t} \gamma_j^{W \neq W_r}(t) > 0.5 \\ \beta, & \text{otherwise} \end{cases}, \tag{54}$$

where Z_{k-1} is to guarantee $\sum_{t=1}^{T} \epsilon_r^k(t) = T$.

8: **end for**

9: Evaluate the class-specific error cost ε_k^j using (49).

10: Train $\mathcal{M}_k$ using boosted Non-uniform MCE with $\epsilon_r^k(t)$

11: **end for**

Output: Generate acoustic score with the combined model $\mathcal{M}$ using (52).

provided. The training utterances are selected through all Switchboard corpus. For the general ASR experiments, we use HUB5 English evaluation (LDC2002S10) as the test set. For keywords spotting experiments, we construct the test set as follows: we first extract the conversations which is on the topic of "CREDIT CARD USE" (this information is not included in the release, but can be downloaded from Switchboard LDC official website) as the test utterances and finally there are 5,649 utterances are selected for testing, note that there are no overlapping utterances with training set as we only use about half data for training. 18 keywords are selected for the spotting evaluation based on their relevance to the topic and occurrence, which are "BANK", "CARD", "CASH", "CHARGE", "CHECK", "MONTH", "ACCOUNT", "BALANCE", "CREDIT", "DOLLAR", "HUNDRED", "LIMIT", "MONEY", "PERCENT", "TWENTY", "VISA", "DISCOVER", "INTEREST" as in [59].

The baseline ASR system is built using Kaldi Speech Recognition Toolkit [60], cross-word triphone models represented by 3-state left-to-right HMMs (5-state HMMs for silence) are trained using MLE on about half the data of whole Switchboard Corpus and a tri-gram language model is trained for decoding. The input features are MFCCs coupled with their linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT) [61] and feature-space maximum likelihood linear regression (fMLLR) [62] for speaker adaptation during later iterations. The WER of the baseline system on HUB5 is 33.4%.

2.5.1.1 Validation of MCE Implementation in the WFST framework with the two improvements

To validate our MCE implementation in the WFST framework with the two improvements described in Section 2.4.1. We first conduct the ASR experiments and compare the WER results of our boosted fundamental MCE with other state-of-arts DT methods (MMI, boosted MMI, MPE). As mentioned earlier, we use HUB5 English evaluation as the test set. For each DT method, we conducted the training in 4 iterations and then we use them to decode

Table 1: WERs of different DT methods on HUB5 English test set

Method	Iteration	WER (LM scale)
MLE (Baseline)	-	33.4% (13)
MMI	4	30.8% (12)
Boosted MMI (b = 0.1)	4	30.6% (12)
MPE	4	30.8% (13)
MCE (boosted)	4	**30.3% (12)**

the utterances in HUB5 with different language model (LM) scales from 9 to 16 respectively. Note that the two improvements we mentioned are also incorporated in MMI and Boosted MMI in our experiments. The boosted factor used in boosted MMI is set to 0.1 and γ and θ in (17) is set to 0.08 and 0 respectively for MCE. For each DT methods, we report best WER among the different LM scales in the form of "WER (LM scales)" in Table 1, which shows our MCE implementation in the WFST framework after two improvements introduced in EBW updates as in Section 2.4.1 can achieve the best word accuracy with 30.3 compared to three other state-of-art DT methods in this task.

2.5.1.2 FOMs results on Credit Card Use subset of Switchboard w.r.t different decaying factor β and initial error cost funtion K_1 and K_2

We conduct both MCE (basic and boosted one with two improvements) and adaptive boosted non-uniform MCE in 4 iterations. We report FOMs w.r.t the different decaying factor β, initial values K_1 and K_2 of the error cost function in Table 2. Note that with all setups of adaptive boosted non-uniform MCE, we floor all values of the error cost function to 1 when decaying the values of the error cost function during each training iteration. It is clear that MCE can be regarded as the case that K_1 and K_2 are both set to 1. As can be seen, the boosted MCE already got 3.40% absolute FOMs improvements over baseline, after embedding different error cost functions, the performance is further improved.The highest FOM is 88.45%, which is 4.86% and 1.46% absolute FOMs improvement over baseline and discriminatively trained system.

Table 2: Keyword spotting evaluations on Credit Card Use subset

Method	K_1	K_2	β	FOM
Baseline	-	-	-	83.59%
MCE	1	1	-	85.34%
MCE (boosted)	1	1	-	86.99%
	2	2	0.3	87.75%
	2	2	0.5	87.77%
	2	2	0.7	87.70%
	3	2.5	0.3	88.04%
	3	2.5	0.5	87.93%
	3	2.5	0.7	87.77%
Adaptive	3	3	0.3	87.89%
	3	3	0.5	87.84%
	3	3	0.7	87.73%
Boosted	4	4	0.3	88.19%
	4	4	0.5	88.08%
	4	4	0.7	88.12%
Non-uniform	5	5	0.3	88.08%
	5	5	0.5	87.96%
	5	5	0.7	87.76%
MCE	6	6	0.3	88.20%
	6	6	0.5	88.33%
	6	6	0.7	88.26%
	7	6	0.3	88.23%
	7	6	0.5	88.15%
	7	6	0.7	88.24%
	7	7	0.3	**88.45%**
	7	7	0.5	88.29%
	7	7	0.7	88.22%

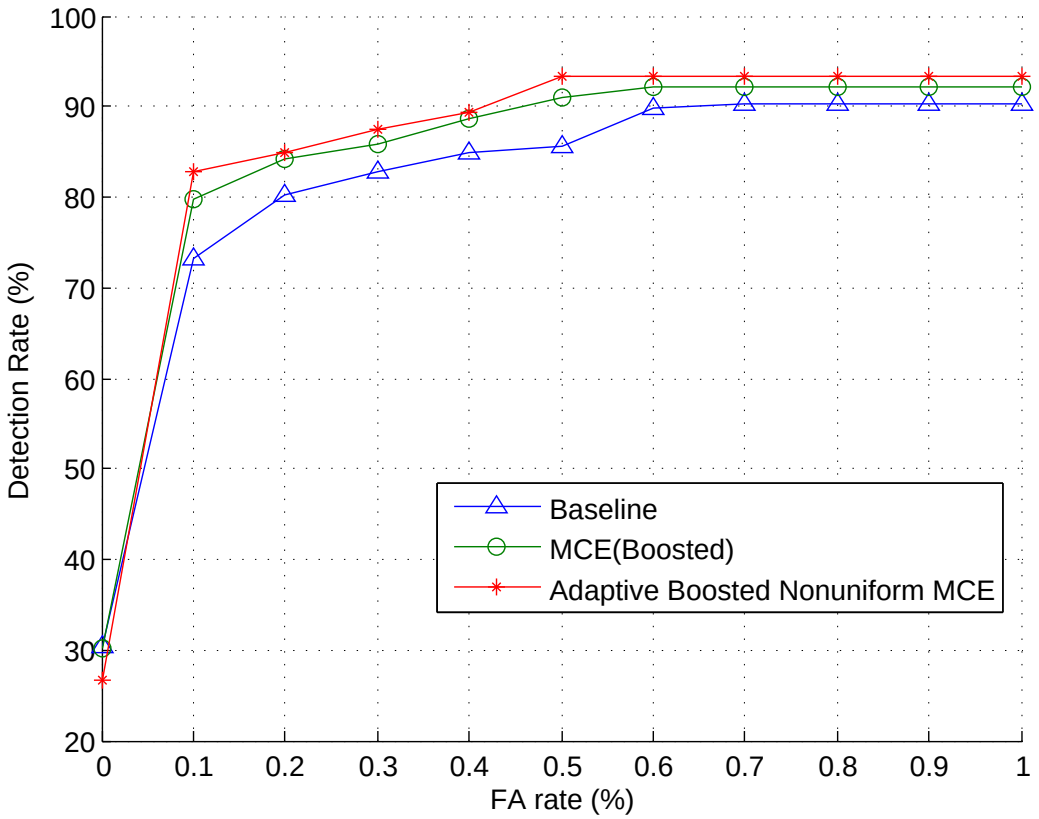

Figure 6: ROC curves on Credit Card Use subset for baseline, discriminatively trained and best performing non-uniform MCE system

2.5.1.3 ROC curves of baseline, discriminatively trained and our system

To show the hit rates at different levels of the false alarm rates, we draw ROC curves for baseline, discriminatively trained (boosted MCE) system and our best performing non-uniform MCE system respectively in Fig.6. As shown in the figure, the adaptive boosted non-uniform MCE achieves consistent and significant FOMs gains over both baseline and discriminatively trained systems at almost all operating points (except 0% false alarms) on the ROC curves.

2.5.1.4 The influence of the initial values K_1 and K_2 in error cost function on spotting performance

To investigate the influence of the initial values in error cost function on spotting performance, we illustrate the FOMs w.r.t. different values of K_1, K_2 from 1 to 7 in Fig 7. Note that the system with $K_1 = K_2 = 1$ is equivalent to MCE with no error cost embedded and

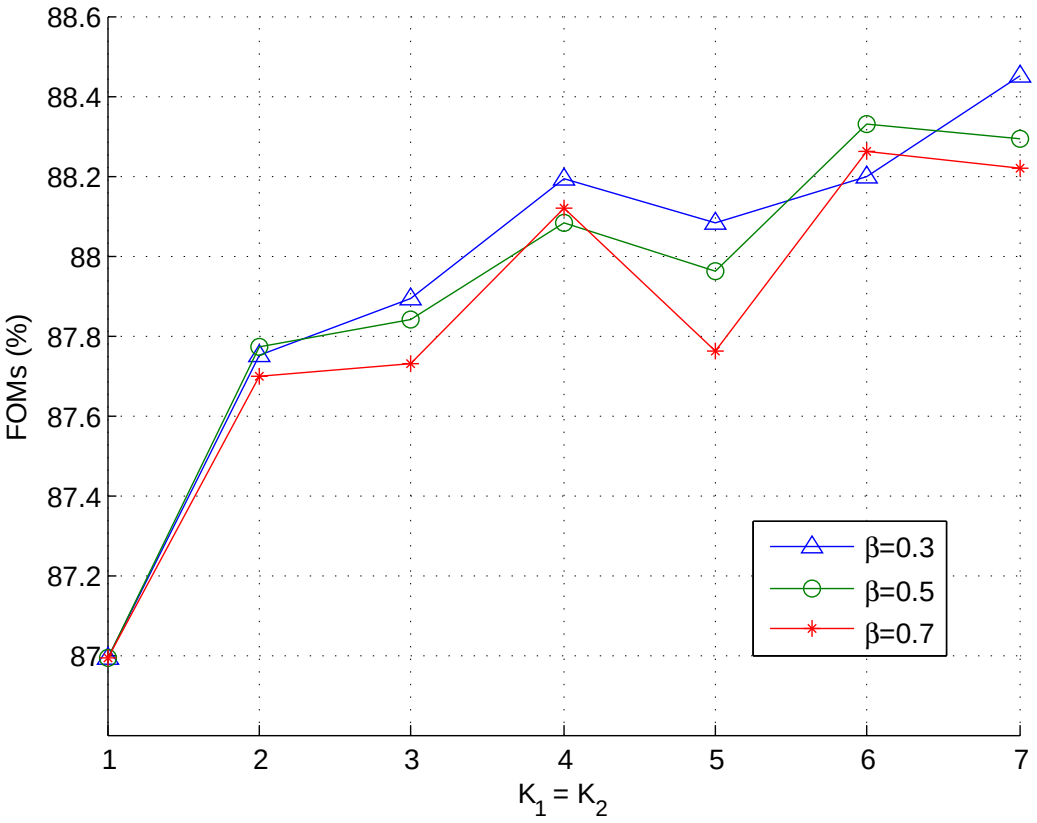

Figure 7: FOMs of the systems with both K_1 and K_2 set from 1 to 7 in three cases: $\beta = 0.3, 0.5, 0.7$

we only illustrate the cases that K_1 is equal to K_2. More details related to the effect of unbalanced values between K_1 and K_2 will be presented in Section 2.5.2.5. From the figure, we can see there is clear tendency that with larger K_1 and K_2 the systems achieves higher FOMs. Another observation is that if we focus the curves from $K_1 = K_2 = 2$ ($K_1 = K_2 = 1$ corresponds to the special case with no error cost function embedded), although the FOMs tend to go higher with the increasing K_1 and K_2, there exists unfavourable FOMs fluctuation when $\beta = 0.7$. However, the fluctuation is obviously mitigated when the decaying factor β is set to a smaller value 0.3 or 0.5. This motivates us to analysis more detailed effect of β on spotting performance in Section 2.5.2.4.

2.5.2 Experiments on HKUST Mandarin Telephone

HKUST mandarin telephone (LDC2005S15) is a 150+ hours of Mandarin Chinese CTS collected by the Hong Kong University of Science and Technology (HKUST), this release contains the training and development sets with 873 and 24 calls respectively.

Since there is no lexicon provided with the corpus and it contains both Chinese and English words (it is highly likely English words occur in spontaneous mandarin speech) below we briefly describe how we prepare the bilingual lexicon. For the Chinese word pronunciations (word to Pinyin), we use one available online dictionary CEDICT [1] for in-vocabulary Chinese words. For OOVs, we do Chinese characters mapping and enumerate all possible pronunciations for each word. We map all Pinyin initials and finals (with tones) to Arpabet phonemes which are widely used in English via IPA rules similar to [63]. For the English word pronunciations, we use CMU dictionary [2] for in-vocabulary words. For OOVs, we use a pre-trained one grapheme to phoneme tools, Sequitur G2P [64]. Since there are several Arpabet phonemes missing for English words pronunciations, what we do is we first mapping the Arpabets to Pinyin (we use the similar mapping rules to [65]), and map them back to Arpabets again but with different phonemes that are within the Arpabet phonemes we use. Finally, a bilingual lexicon is built based on a unified phoneme set. And for each phoneme (except those corresponding to Pinyin initials since there are no tones for them), we have 6 tones, eg., AO (mainly used for English words), AO1, AO2, AO3, AO4, AO5. We let each phoneme with the different tones to share the same root in the decision tree while making extra tonal questions for them.

We use an open-source tools mmseg [3] to do the Chinese word segmentation and then a tri-gram language model is trained on all transcriptions from training set.

For other components of ASR experiments setups, they are similar to what is described

[1] available on `http://www.mdbg.net/chindict/chindict.php?page=cedict`
[2] available on `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`
[3] available on `http://pypi.python.org/pypi/mmseg/1.3.0`

Table 3: CERs of different DT methods on HKUST Mandarin Telephone Dev Set

Method	Iteration	CER (LM scale)
MLE (Baseline)	-	49.67% (13)
Boosted MMI (b = 0.1)	4	**44.24% (11)**
MPE	4	44.96% (12)
MCE (boosted)	4	44.74% (11)

in Section 2.5.1. The *character error rate* (CER) of the baseline system on the development set is 49.67%, which is comparable to the results reported in [66]. For the keywords spotting evaluations, we use the development set as the test set and 20 Chinese keywords are selected: 喜欢 (like), 中国 (China), 大学 (university), 生活 (life), 朋友 (friend), 国家 (country), 足球 (football), 黄山 (Huangshan), 锻炼 (exercise), 篮球 (basketball), 唱歌 (sing),工作 (job), 专业 (major), 运动 (sports), 电视 (televisions), 体育 (sports), 学习 (study), 问题 (problem), 台湾 (Taiwan), 学生 (student).

2.5.2.1 CERs results on development set of HKUST Mandarin Telephone using different DT methods

We first conduct the ASR experiments and compare the CER results of our boosted fundamental MCE with two other state-of-arts DT methods, boosted MMI and MPE. We use the HKUST Mandarin Telephone released development set of as the test set. For each DT methods, we conducted the training in 4 iterations and decoding with different language model (LM) scales from 9 to 20 respectively. The setups of three DT methods are similar to Section 2.5.1.1. For each DT methods, we report best CER among the different LM scales in the form of "WER (LM scales)" in Table 3. As can be seen in the table, our MCE implementation in the WFST framework after two improvements introduced in EBW updates as in Section 2.4.1 can achieve comparable character accuracy with both Boosted MMI and MPE.

2.5.2.2 *FOMs results on development set of HKUST Mandarin Telephone w.r.t different decaying factor β and initial error cost funtion K_1 and K_2*

We first conduct keywords spotting experiments with MLE baseline and adaptive boosted non-uniform MCE systems and we find interesting FOMs results, which are shown in the first four rows of Table 4 that the FOM with MCE system is even slightly worse than one got from MLE baseline system. For verifications, we further conducted spotting experiments with other two DT methods, Boosted MMI and MPE. The results show that although those systems of these fundamental DT methods can achieve significant character accuracy gains in general as in Section 2.5.2.1, when it comes to the FOMs, which measure the performance by taking into accounts both the accuracies and false alarms only on a set of keywords, the fundamental DT methods fail to reduce two type of errors on the keywords effectively. This substantially illustrates the advantage of our non-uniform MCE. Similarly, we report FOMs of our adaptive boosted non-uniform MCE systems w.r.t the different decaying factor β, initial values K_1 and K_2 of the error cost function in Table 4. As can be seen, even when the fundamental DT methods fail to give the consistent spotting performance gain, our systems achieves consistent and significant performance gains over both ML baseline and discriminatively trained systems. The highest FOMs is 61.57%, which is almost around 4.0% absolute FOMs improvement over both baseline and all discriminatively trained systems.

2.5.2.3 *ROC curves of baseline, discriminatively trained and our system*

To show the hit rates at different levels of the false alarm rates, we also draw ROC curves for baseline, discriminatively trained (MPE) system and our best performing non-uniform MCE system respectively on this tasks in Fig.8. As shown in the figure, the adaptive boosted non-uniform MCE achieves consistent and significant hit rates gains over both baseline and discriminatively trained systems at almost all operating points (except 1 false alarms) on the ROC curves.

Table 4: Keyword spotting evaluations on the Dev Set of HKUST Mandarin telephone speech

Method	K_1	K_2	β	FOM
MLE	-	-	-	57.19%
Boosted MMI	-	-	-	56.86%
MPE	-	-	-	59.11%
MCE (boosted)	1	1	-	57.14%
Adaptive	3	3	0.3	60.22%
	3	3	0.5	60.15%
	3	3	0.7	59.94%
	4	4	0.3	59.77%
	4	4	0.5	60.12%
	4	4	0.7	59.64%
	5	5	0.3	60.73%
	5	5	0.5	60.80%
	5	5	0.7	59.39%
Boosted	6	6	0.3	60.40%
	6	6	0.5	60.32%
	6	6	0.7	60.54%
Non-uniform	6	5	0.3	60.46%
	6	5	0.5	60.95%
	6	5	0.7	60.21%
MCE	7	7	0.3	**61.57%**
	7	7	0.5	60.77%
	7	7	0.7	59.90%
	7	6	0.3	61.40%
	7	6	0.5	60.59%
	7	6	0.7	60.37%
	8	8	0.3	61.37%
	8	8	0.5	61.21%
	8	8	0.7	60.75%
	8	7	0.3	61.42%
	8	7	0.5	61.17%
	8	7	0.7	61.12%

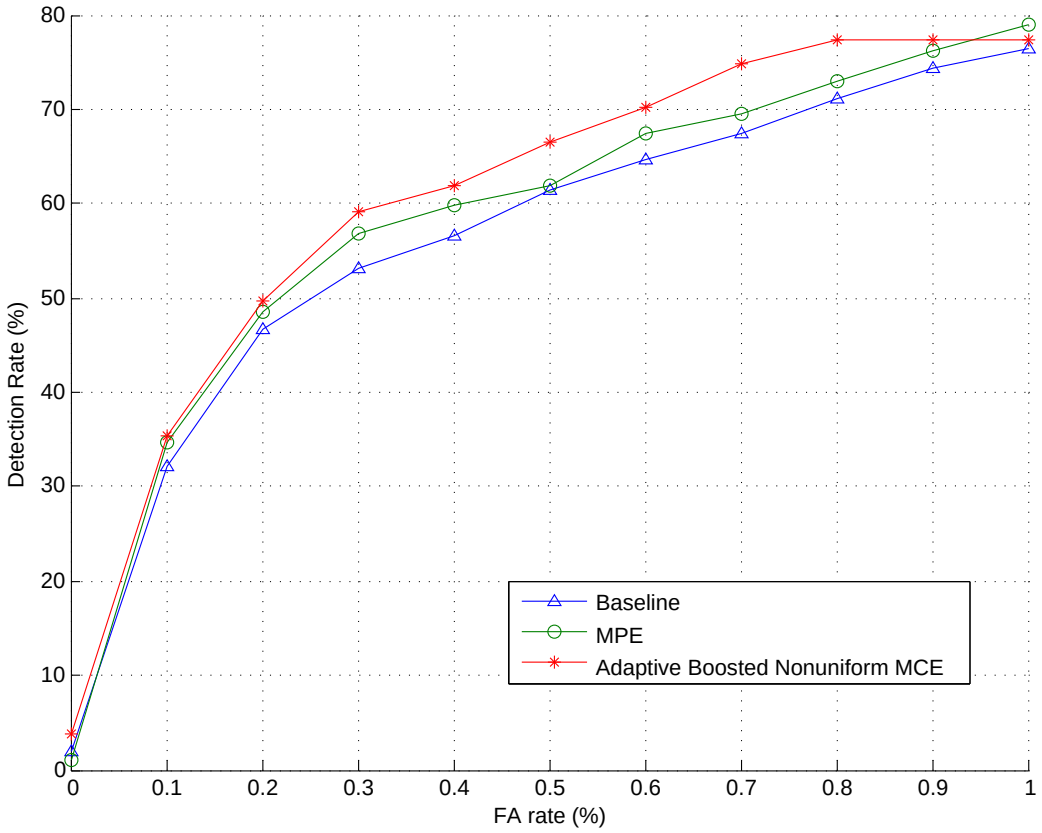

Figure 8: ROC curves on Dev Set of HKUST Mandarin telephone speech for baseline, discriminatively trained and best performing non-uniform MCE system

2.5.2.4 The influence of decaying factor β in error cost function on spotting performance

To investigate the more detailed influence of the decaying factor β for the adaptive adjustment of the error cost function on spotting performance, we use smaller step size with 0.5 to increasing K_2 and use step size 1 to increase K_1, and during different trials we only iterate all the cases that $K_1 - 1 \leq K_2 \leq K_1$, *e.g.*, the values of (K_1, K_2) could be the sequence of $(3,3)$, $(4,3)$, $(4,3.5)$, $(4,4)$, $(5,4)$... We draw the FOMs of each trials with the sequence of the different combinations of K_1, K_2 staring from $K_1 = K_2 = 3$ to $K_1 = K_2 = 8$ (note that the trial 0 is the boosted MCE with no error cost function embedded) in four different different decaying factor $\beta = 0.3, 0.5, 0.7, 1.0$ in Fig 9. (Note that when $\beta = 1.0$, there is no adaptive adjustment of the error cost function at all). From the figure, it is also can be seen that FOMs tend to go higher with larger K_1 and K_2 in all cases. For the curves of different decaying factor β, it is clear that the non-uniform MCE with the adaptive error

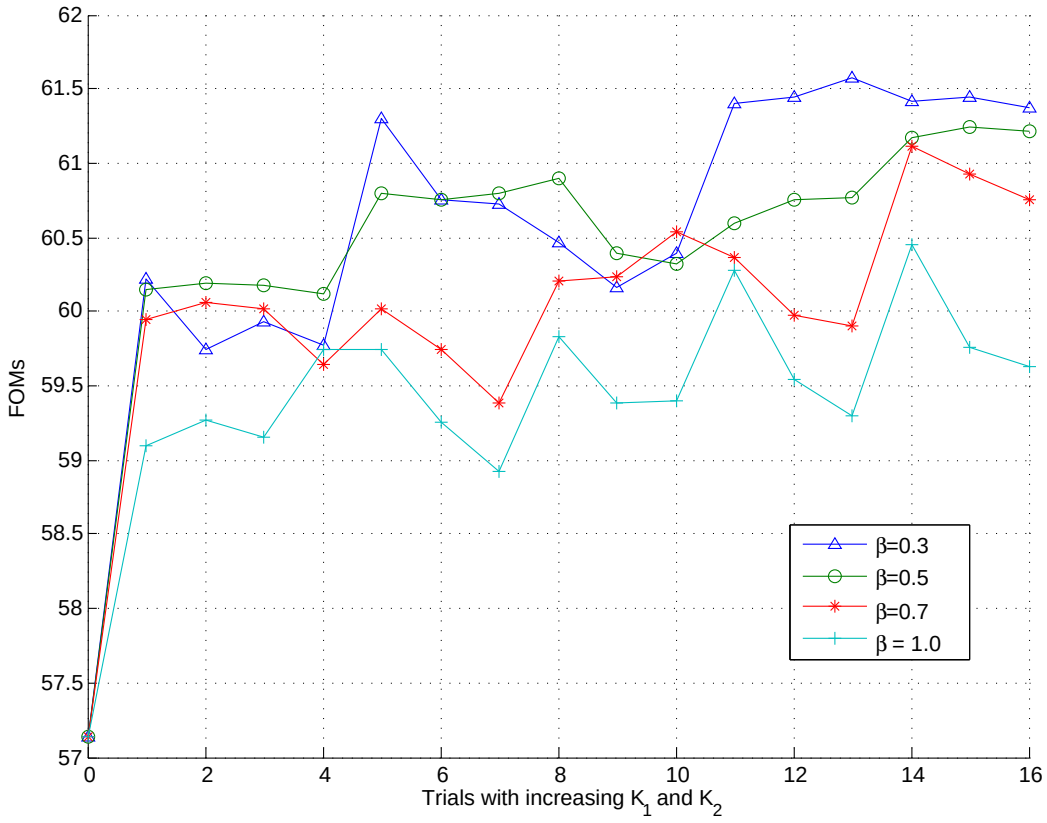

Figure 9: FOMs of different keyword spotting trials with increasing K_1 and K_2 in four different decaying factor $\beta = 0.3, 0.5, 0.7, 1.0$ on Dev Set of HKUST Mandarin telephone speech, note that $K_1 = K_1 + 1$, $K_2 = K_2 + 0.5$, $K_1 - 1 \leq K_2 \leq K_1$.

cost decaying ($\beta = 0.3, 0.5, 0.7$) performs consistently better than the one without ($\beta = 1$). Another observation is that there exists all unfavourable FOMs fluctuations when $\beta = 0.3$, $\beta = 0.7$ and $\beta = 1.0$, however, the fluctuation in the case of $\beta = 0.3$ is dampened when K_1 and K_2 go fairly large but the ones when $\beta = 0.7$ and $\beta = 1.0$ are not. And this fluctuations seems not so severe in the case of $\beta = 0.5$.

To gain an insight of the significance of the adaptive adjustment for the error cost functions, we list several typical FOMs of non-uniform MCE with and without the adaptive decaying schemes in Table 5 for the comparisons. From the table, we can see there exists considerable absolute FOMs difference between two cases from 1.12% to 2.27% and with larger K_1 and K_2, the effect of the adaptive error cost adjustment scheme becomes

Table 5: FOMs comparisons between non-uniform MCE with and without the scheme of adaptive error cost function embedding, $\beta = 1$ corresponds to the case without adaptive adjustment of error cost.

K_1	K_2	β	FOMs	Improvements
3	3	1	59.10%	-
3	3	0.3	60.22%	1.12%
5	4	1	59.74%	-
5	4	0.3	61.03%	1.29%
5	4.5	1	59.26%	-
5	4.5	0.3	60.76%	1.5%
7	6.5	1	59.55%	-
7	6.5	0.3	61.44%	1.89%
7	7	1	59.30%	-
7	7	0.3	61.57%	**2.27%**

more significant which is in line with our expectation since larger error cost function embedding would lead to higher risk of over-training while the introduced adaptive error cost adjustment scheme aims to avoid it.

2.5.2.5 *The trade-off between K_1 (hits) and K_2 (false alarms) in error cost function on spotting performance*

As aforementioned, one can design the error cost function in a asymmetrical way, *i.e.*, using different values for K_1 and K_2 to achieve desirable compromise between the detection miss and false alarm rate. To investigate this, we list number of hits and false alarms (FAs) of several cases when K_1 and K_2 are assigned different values with the same β in Table 6. From the table, when the difference between K_1 and K_2 is significant enough compared to the their values, *e.g.*, when $K_1 = 4, 5, 6$ and $K_1 - K_2 = 1$, we can observe there are considerable false alarms reduction and almost the same hit rate when we enlarge the value of K_2 appropriately. But when the difference ($K_1 - K_2 = 0.5$) is small and the values of K_1 and K_2 are fairly large ($K_1 = 7, 8$), the effect begins to disappear.

Table 6: Using different K_1 and K_2 to achieve the compromise between hits and false alarms rate: the FOMs w.r.t the unbalanced values of K_1 and K_2 with same β.

K_1	K_2	β	# Hits	# FAs
4	3	0.3	764	215
4	3.5	0.3	765	210
5	4	0.3	763	204
5	5	0.3	763	202
6	5	0.3	760	207
6	6	0.3	760	203
7	6.5	0.5	763	203
7	7	0.5	762	202
8	7	0.5	762	188
8	7.5	0.5	760	186
8	8	0.5	760	187

2.5.3 Discussions

It can be seen from all the results we reported above on two large-scale CTS tasks that our non-uniform MCE framework can achieve notable and consistent spotting performance gains over both MLE baseline and discriminatively trained systems. However, to automatically determine the best combinations of K_1, K_2 and β is not trivial since it is also related to the dispersion characteristics of keywords in certain training set, thus merits further investigations. We already observed the significant influence of β on the spotting performance. This motivates us to also allow the adaptive adjustment of the decaying factor β, *i.e.*, the speed for decaying the error cost. And one possible way is to tweak β according to the normalization factor Z_k as in (54) of each non-uniform MCE iteration, which to some extent can monitor the training iterations to prevent them from over-training. We will leave more details related to this in the future work.

2.6 Summary

In this chapter, we present a complete framework of DT using non-uniform criteria for keyword spotting. First, we generalized keyword spotting as a non-uniform error ASR problem, and then proposed and formulated non-uniform MCE for keyword spotting. The

main idea is to adapt the fundamental MCE criteria in a cost-sensitive way which leads optimizations to place emphasis on keywords. We also present an ingenious way which takes advantage of WFST's difference operations to efficiently implement the non-uniform MCE in the WFST framework. To further boost the spotting performance and tackle the potential issue of over-training in the non-uniform MCE, we present a complete framework, adaptive boosted non-uniform MCE for keyword spotting. We first make two improvements to the fundamental MCE optimization procedure. On the top of this boosted MCE and motivated by AdaBoost [46], we introduce an adaptive scheme to embed error cost functions, namely the adaptive adjustment of the error cost function depending on whether current frame is classified correctly or not, together with the model combinations during the decoding procedure.

We comprehensively validate the proposed framework on two challenging large-scale spontaneous conversational telephone speech (CTS) tasks in different languages (English and Mandarin), Switchboard and HKUST Mandarin telephone speech. The experiments results showed our framework can achieve consistent and significant spotting performance gains over both MLE baseline and discriminatively trained systems. We also investigate the influences of different setups (K_1, K_2, β) in our framework on spotting performance. It is observed that with increasing values of the initial values K_1 dn K_2 in the error cost function, the spotting performance is further improved accordingly, though there exists FOMs fluctuations when β is set to relative large values. In addition, it is clear that with smaller decaying factor β, the spotting performance tend to go higher. Finally, we briefly investigate using different values of K_1 and K_2 to achieve the desirable compromise between hit and false alarm rate.

In the future work, motived by the significant influence of decaying factor β on spotting performance observed in the experiments, we will introduce more complete adaptive adjustment scheme to allow β be adaptively tweaked rather than a fixed value during iterations and also different significant level among keyword frames into the framework.

CHAPTER 3

RECURRENT DEEP NEURAL NETWORKS FOR ROBUST SPEECH RECOGNITION

3.1 Introduction

Improving environmental robustness of automatic speech recognition (ASR) systems has been studied for decades. To deal with the mismatched acoustical conditions between training and testing, feature space compensation approaches typically involve removing additive noise and channel distortions using speech enhancement techniques [7] such as spectral subtraction, Wiener filtering and MMSE estimators [67, 68, 69]. Other researchers explored use of noise resistant features [9, 70] or feature transformations [71, 72]. Model adaptation methods attempt to achieve compensation by adapting the models to the noisy condition. The most straightforward way is using the multi-style training strategy [73] to train models on the multi-condition data that includes different acoustical conditions of the test data. Other model space adaptation methods include parallel model combination (PMC), data-driven PMC [74] and vector Taylor series (VTS) based compensation [75, 76, 77]. The combination of both feature space and model space compensation techniques usually offer the state-of-the-art environmental robustness for an ASR system.

Recently, deep neural network (DNN) based acoustic models have been introduced for LVCSR [78, 31] tasks and show its great success in both Tandem [28] and hybrid DNN-HMM systems [30]. This opens new possibilities for further improving the noise robustness of ASR systems. In [79] and [80], it is shown that DNN based systems have remarkable robustness to environment distortions and the authors can achieve state-of-the-art performance on Aurora-4 benchmark without multiple decoding passes and model adaptation. Meanwhile, recurrent neural networks (RNNs) have been also explored for robust ASR in [81, 82, 83, 84]. However, the authors only investigated RNNs in the Tandem setup or used it as a front-end denoiser and reported results on a small vocabulary task. Few if any have

explored the RNNs combined with deep structure in the hybrid setup and report results on larger tasks where the language model (LM) matters during decoding.

In this chapter, we investigate the RNNs with deep architecture in hybrid systems for robust ASR. Specifically, we add full recurrent connections to certain hidden layer of a feedforward DNN to allow the model to capture the temporal dependency in deep representations. A new backpropagation through time (BPTT) algorithm for updating the parameters of the recurrent layer is introduced to make the minibatch stochastic gradient descent (SGD) on the proposed recurrent DNN more efficient and effective. We evaluate the proposed recurrent DNN architecture under the hybrid setup on both the 2nd CHiME challenge (track 2) [85] and Aurora-4 tasks. Experimental results on the CHiME challenge data show that we can obtain consistent 7% relative WER improvements over DNN systems, achieving the state-of-the-art performance reported in [86] without front-end preprocessing, speaker adaptive training and multiple decoding passes. For the experiments on Aurora-4, the proposed system achieves 4% relative WER improvement over a strong DNN baseline system.

The remainder of this chapter is organized as follows. In Section 3.2, we review the DNN-HMM hybrid system and describe the architecture of the recurrent DNN. A new backpropagation through time algorithm for the recurrent layer and minibatch SGD on the whole network will be elaborated in Section 3.3. We report our experimental results in Section 3.4 and conclude our work in Section 3.5.

3.2 Recurrent DNN Architecture

3.2.1 Hybrid DNN-HMM System

In a conventional GMM-HMM LVCSR system, the state emission log-likelihood of the observation feature vector $\mathbf{o}_t$ for certain tied state or senone s_j of HMMs is generated using,

$$\log p(\mathbf{o}_t|s_j) = \log \sum_{m=1}^{M} \pi_{jm} \mathcal{N}_{jm}(\mathbf{o}_t|s_j), \tag{55}$$

where M is the number of Gaussian mixtures in the GMM for state j and π_{jm} is the mixing weight. As the outputs from DNNs represent the state posteriors $p(s_j|\mathbf{o}_t)$, a DNN-HMM hybrid system [31] uses pseudo log-likelihood as the state emissions,

$$\log p(\mathbf{o}_t|s_j) \propto \log p(s_j|\mathbf{o}_t) - \log p(s_j), \tag{56}$$

where the state priors $\log p(s_j)$ can be estimated using the state alignments on the training speech data. The input features vectors $\mathbf{o}_t$ to the first layer of DNNs usually use a context of l frames [31], *e.g.*, $l = 9$ or $l = 11$.

3.2.2 Recurrent Deep Architecture

The architecture of recurrent DNN we use is shown in Fig.10. The fundamental structure is a feedforward DNN but with certain hidden layer having full recurrent connections with itself (In the Fig.10, the third hidden layer from the input layer has recurrent property). The values corresponding to those neurons at the feedforward hidden layers can be expressed as,

$$\mathbf{x}^i = \begin{cases} W_1\mathbf{x}^0 + \mathbf{b}_1, & i = 1 \\ W_i\mathbf{y}^{i-1} + \mathbf{b}_i, & i > 1 \end{cases}, \tag{57}$$

$$\mathbf{y}^i = \begin{cases} \text{sigmoid}(\mathbf{x}^i) & i < n \\ \text{softmax}(\mathbf{x}^i) & i = n \end{cases}, \tag{58}$$

where n is the total number of the feedforward hidden layers and both the sigmoid and softmax functions are element-wise operations. The vector $\mathbf{x}^i$ corresponds to pre-nonlinearity activations except that $\mathbf{x}^0$ is the input feature vector and $\mathbf{y}^i$ is the neuron vector at the i^{th} hidden layer. For the recurrent hidden layer, denote by $\mathbf{x}_t^i$ and $\mathbf{y}_t^i$ the pre-nonlinearity activation vector and neuron vector at frame t, the value of neuron vector at the i^{th} hidden layer is given by,

$$\mathbf{x}_t^i = W_{ii}\mathbf{y}_{t-1}^i + \mathbf{b}_{ii} + W_i\mathbf{y}_t^{i-1} + \mathbf{b}_i \tag{59}$$

$$\mathbf{y}_t^i = \text{sigmoid}(\mathbf{x}_t^i), \tag{60}$$

where W_{ii} and $\mathbf{b}_{ii}$ are the recurrent weight matrix and bias vector.

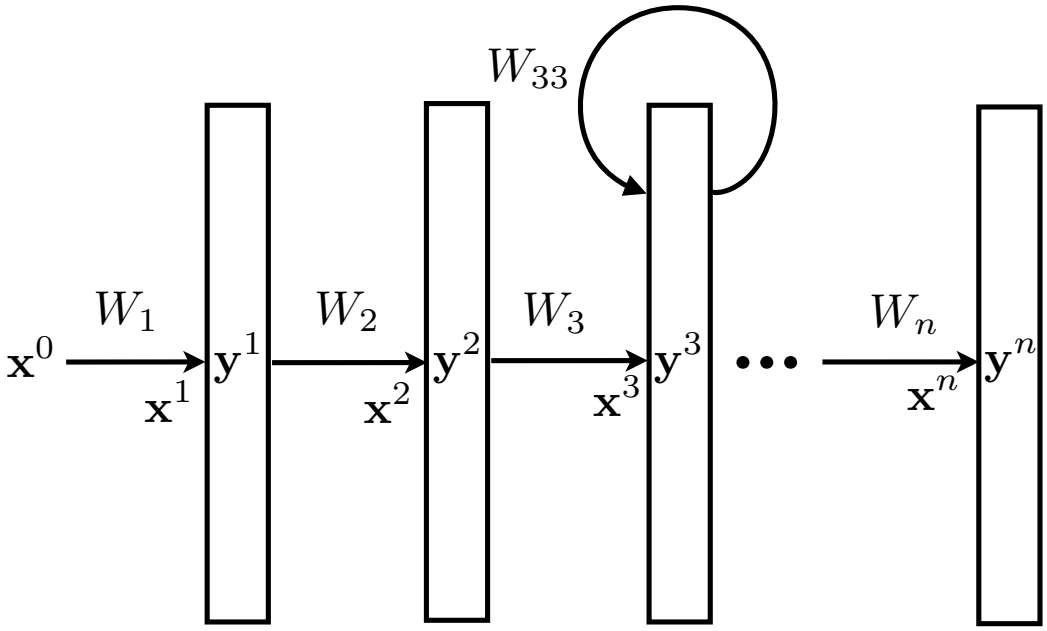

Figure 10: Recurrent DNNs architecture: the third layer from the input layer is the recurrent hidden layer with the parameters W_{33}, note that the bias terms are omitted for simplicity.

3.3 Backpropagation on the recurrent DNN

3.3.1 Backpropagation on the Feedforward Layers

For convenience, we will use the notations as shown in Fig.10. Taking partial derivatives of the loss objective function with respect to the pre-nonlinearity activations of output layer ($\mathbf{x}^n$ in the Fig.10) will give us the error vector to be backpropagated to the previous hidden layers. The negative cross-entropy is commonly used loss function. The loss functions based on discriminative training criteria such as sMBR [87], MMI and MPE/MWE [88] have also been used for ASR. When various loss functions are used, the only difference reflected in the backpropagation lies in the error vector we backpropagate to the previous hidden layers. If we use the negative cross-entropy loss and let $\mathcal{X}$ be the whole training set which contains N frames, *i.e.*, $\mathbf{x}^0_{1:N} \in \mathcal{X}$, then the loss associated with $\mathcal{X}$ is given by,

$$\mathcal{L}_{1:N} = -\sum_{t=1}^{N}\sum_{j=1}^{J} \mathbf{d}_t(j) \log \mathbf{y}^n_t(j), \tag{61}$$

and $\mathbf{d}_t(j)$ is the j^{th} element of the label vector at frame t, then the error vector to be back-propagated to the previous layers is given by,

$$\boldsymbol{\epsilon}_t^n = \frac{\partial \mathcal{L}_{1:N}}{\partial \mathbf{x}^n} = \mathbf{y}_t^n - \mathbf{d}_t, \tag{62}$$

the backpropagated error vectors at previous hidden layer are thus,

$$\boldsymbol{\epsilon}_t^i = W_{i+1}^T \boldsymbol{\epsilon}_t^{i+1} * \mathbf{y}^i * \left(\mathbf{1} - \mathbf{y}^i\right), i < n \tag{63}$$

where $*$ denotes element-wise multiplication. With the error vectors at certain hidden layers, the gradient over the whole training set with respect to the weight matrix W_i is given by,

$$\frac{\partial \mathcal{L}_{1:N}}{\partial W_i} = \mathbf{y}_{1:N}^{i-1} (\boldsymbol{\epsilon}_{1:N}^i)^T, \tag{64}$$

note that in above equation, both $\mathbf{y}_{1:N}^{i-1}$ and $\boldsymbol{\epsilon}_{1:N}^i$ are matrices, which is formed by concatenating vectors corresponding to all the training frames from frame 1 to N, *i.e.*, $\boldsymbol{\epsilon}_{1:N}^i = [\boldsymbol{\epsilon}_1^i, \ldots, \boldsymbol{\epsilon}_t^i, \ldots, \boldsymbol{\epsilon}_N^i]$. The batch gradient descent updates the parameters with the gradient in (64) only once after each sweep through the whole training set and in this way parallelization can be easily conducted to speedup the learning process. However, SGD usually works better in practice where the true gradient is approximated by the gradient at a single frame t, *i.e.*, $\mathbf{y}_t^{i-1}(\boldsymbol{\epsilon}_t^i)^T$, and the parameters are updated right after seeing each frame. The compromise between the two, minibatch SGD, is more widely used, as the reasonable size of minibatches makes all the matrices fit into GPU memory, which leads to a more computationally efficient learning process.

3.3.2 BPTT on the Recurrent Layer

BPTT updates the recurrent weights by unfolding the networks in time. As shown in Fig.11, the standard error BPTT over a minibatch $\mathbf{x}_{1:M}^0 \in \mathcal{X}$ is given by,

$$\boldsymbol{\epsilon}_t^i = \begin{cases} (W_{i+1}^T \boldsymbol{\epsilon}_t^{i+1}) * \mathbf{y}_t^i * \left(\mathbf{1} - \mathbf{y}_t^i\right), & t = M \\ (W_{i+1}^T \boldsymbol{\epsilon}_t^{i+1} + W_{ii}^T \boldsymbol{\epsilon}_{t+1}^i) * \mathbf{y}_t^i * \left(\mathbf{1} - \mathbf{y}_t^i\right), & t < M \end{cases}, \tag{65}$$

where M is the size of minibatch. Note that the evaluation of the exact gradients with respect to the recurrent weight matrix W_{ii} needs the corresponding error vectors backpropagated through time to the first frame of the current training speech utterance, but in practice the gradients with respect to the recurrent weight matrix W_{ii} over the minibatch are usually approximated by truncating the BPTT process within the corresponding minibatch. In (65), each time step of BPTT needs both the error vector from next frame and the one

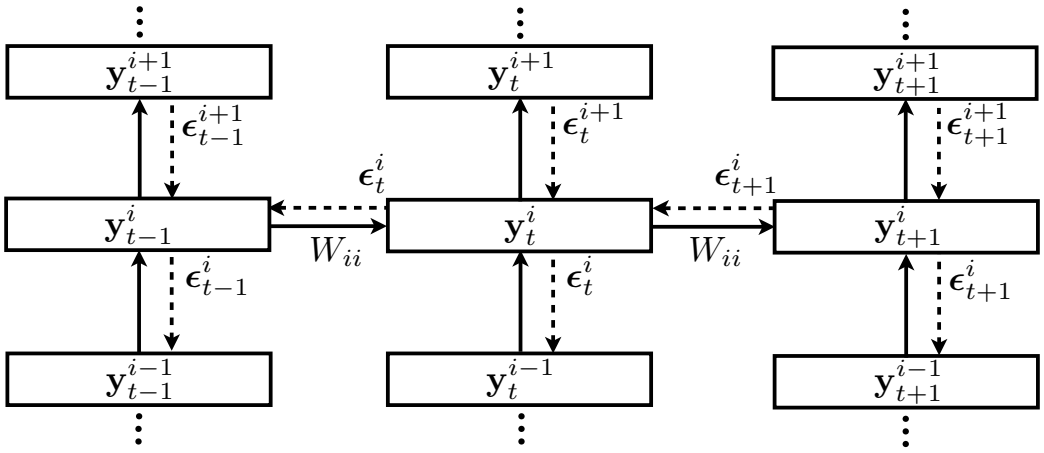

Figure 11: Backpropagation through time for i^{th} recurrent layer's parameter W_{ii}: the solid lines denote the directions of forward propagation and the dotted lines denote the directions of backpropagation.

from next hidden layer which forces us to backpropagate the error vector frame by frame rather than in a minibatch mode. In the standard minibatch BPTT, a key observation is that error signals backpropagated from the next hidden layer (*i.e.*, $\epsilon_{t-1}^{i+1}, \epsilon_{t}^{i+1}, \epsilon_{t+1}^{i+1}$ as in Fig.11) will be backpropagated in different time steps which indicates each training frame within a minibatch will make non-uniform contribution to the final minibatch gradient. Thus, we introduce the *truncated minibatch BPTT* where for each individual online gradient, we truncated the BPTT process in fixed time steps,

$$\frac{\partial \mathcal{L}_{1:M}}{\partial W_{ii}} = \sum_{t=1}^{M} \frac{\partial \mathcal{L}_t}{\partial W_{ii}} \approx \sum_{t=1}^{M} \sum_{\tau=1}^{\mathrm{T}} \mathbf{y}_{t-\tau}^{i} (\boldsymbol{\epsilon}_{t-\tau+1}^{i})^T, \tag{66}$$

where $\frac{\partial \mathcal{L}_t}{\partial W_{ii}}$ is the online gradient at frame t, while for each individual online gradient, we backpropagate for multiple time steps as in [22], *e.g.*, T = 4 or T = 5,

$$\boldsymbol{\epsilon}_{t-1}^{i} = W_{ii}^{T}\boldsymbol{\epsilon}_{t}^{i} * \mathbf{y}_{t-1}^{i} * \left(\mathbf{1} - \mathbf{y}_{t-1}^{i}\right). \tag{67}$$

Another benefit of the introduced truncated minibatch BPTT is that after we exchange the summation order (see below), the BPTT on the recurrent weights can be conducted in a minibatch mode,

$$\begin{aligned}\frac{\partial \mathcal{L}_{1:M}}{\partial W_{ii}} &= \sum_{t=1}^{M}\frac{\partial \mathcal{L}_t}{\partial W_{ii}} \approx \sum_{t=1}^{M}\sum_{\tau=1}^{\mathrm{T}}\mathbf{y}_{t-\tau}^{i}(\boldsymbol{\epsilon}_{t-\tau+1}^{i})^{T} \\ &= \sum_{\tau=1}^{\mathrm{T}}\sum_{t=1}^{M}\mathbf{y}_{t-\tau}^{i}(\boldsymbol{\epsilon}_{t-\tau+1}^{i})^{T} \\ &= \sum_{\tau=1}^{\mathrm{T}}\underbrace{\mathbf{y}_{1-\tau:M-\tau}^{i}(\boldsymbol{\epsilon}_{1-\tau+1:M-\tau+1}^{i})^{T}}_{\text{minibatch gradient}},\end{aligned} \tag{68}$$

note that in above equations the vectors with negative indices come from the corresponding ones in previous minibatch. Therefore, the gradients for updating the recurrent weights can also be calculated in a minibatch mode using matrix multiplication which can be considerably speeded up using GPU.

3.4 Experiments

3.4.1 Experiments on CHiME challenge data

Track 2 of the 2nd CHiME challenge [85] is a medium vocabulary (5k) task under reverberated and noisy environment. There are three sets of data, clean, reverberated, isolated (reverberated and noisy). All the clean speech utterances are extracted from WSJ0 database. The reverberated speech utterances are generated by convolving clean speech with time-varying binaural room impulse responses. Noise backgrounds including concurrent speakers, TV, game console, footsteps, and distant noise from outside or from the kitchen are first recorded and the isolated speech utterances are created by selecting appropriate pre-recorded noise background excerpts, mixing them to reverberated speech utterances to obtain the speech signals with the SNR of -6, -3, 0, 3, 6, and 9 dB without

rescaling. The multi-condition training set contains 7138 speech utterances (reverberated and noisy version of SI-84) with SNR from -6 to 9 dB. The development set contains 2460 multi-condition speech utterances and evaluation set contains 1980 (reverberated and noisy version of NOV-92, *i.e.*, 330×6) utterances with uniform number for each condition.

We first build a GMM-HMM system using Kaldi toolkit [60] for the task: 2008 distinct tied-state GMMs are trained with MFCC features coupled with their linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT) on the 7138 multi-condition speech utterances and feature-space maximum likelihood linear regression (fMLLR) for speaker adaptation during later iterations. For the DNN-HMM systems, we first do generative pre-training using RBMs, and stacking them together in the end to initialize the DNN with 7 2048-dim hidden layers. With the alignments obtained from the GMM-HMM system, we train DNN I system with 40-dim log Mel filter-bank features. We use 256 minibatch and 0.008 as the initial learning rate. After each epoch of training, we validate the frame accuracy on the development set, shrink the learning rate by 0.5 when the improvements are less than 0.5% and stop training when the improvements are less than 0.1%. With the trained DNN I system, we do the realignments and train a second DNN system DNN II using the new alignments. We repeat this process until the performance gain from the realignments become saturated. The standard 5k tri-gram language models are used for the decoding. The WER results are listed in Table 7. As can be seen, all DNN systems achieve significant gains over GMM-HMM system, the performance gain from the realignments saturated until the fourth system is trained and the best realigned DNN IV system obtains 29.89% WER, which will be the baseline system we use for the comparison with recurrent DNN system.

Then we build the proposed recurrent DNN-HMM systems. For the comparisons, the alignments used to train all the recurrent DNN system are the same as the DNN IV systems. We initialize recurrent DNN parameters as follows: for the feedforward layers, we just copy the weights from the DNN trained after 5 epochs in DNN IV systems (in our experiment,

Table 7: WERs (%) of baseline GMM-HMM, and DNN-HMM systems on CHiME challenge data, DNN I~IV systems correspond to the iteratively retrained DNNs with the new alignments

Systems	Conditions						Avg.
	9dB	6dB	3dB	0dB	-3dB	-6dB	
GMM	29.87	36.26	43.25	54.64	61.63	69.51	49.19
DNN I	19.19	23.41	28.17	36.56	45.99	56.81	35.02
DNN II	17.88	21.58	24.83	33.42	40.91	52.06	31.78
DNN III	**16.85**	**20.25**	23.05	30.81	39.59	50.70	30.21
DNN IV	16.89	20.29	**22.83**	**30.36**	**39.49**	**49.47**	**29.89**

$15 \sim 17$ iterations are needed to reach convergence. This is for speeding up the training, in the end, the total epochs are almost the same); the recurrent parameters are initialized with randomization. We use 256 minibatch for SGD, and 0.004 for the initial learning rate. The learning rate scheduling and stop criteria are the same as DNN training as described earlier. We try 5 different setups in our recurrent DNN experiments: RDNN system corresponds to the recurrent DNN with the recurrent units at the 4^{th} hidden layer from the input layer using standard minibatch BPTT. RDNN I~IV systems correspond to the recurrent DNN with the recurrent units at the 2^{nd}, 3^{rd}, 4^{th} and 5^{th} hidden layer from the input layer using the introduced truncated minibatch BPTT as described in Section 3.3.2. As shown in Table 8, with the standard minibatch BPTT, system RDNN only shows marginal WER improvements over the baseline DNN system. While with the truncated minibatch BPTT, all recurrent DNN systems significantly outperform the DNN systems in all conditions. This is very likely because the non-recurrent weights are copied from the DNN system while the recurrent weights are randomly initialized. Thus the recurrent weights are less trained if not using the proposed approach. Therefore, the truncated minibatch BPTT will be used through all following recurrent DNN experiments. The best system with recurrent hidden layer at the 3^{rd} layer obtain 27.70% WER, achieving the state-of-the-art performance[1] and

[1]The state-of-the-art system reported in [86] achieves 26.86% WER but with discriminatively trained LM and MBR decoding. With the tri-gram LMs, the best system reported by authors is 27.61%

Table 8: WERs (%) of best DNN-HMM system and five recurrent DNN-HMM systems trained on CHiME challenge multi-condition data: RDNN system corresponds to the recurrent DNN with the recurrent units at 4th hidden layer using standard minibatch BPTT. RDNN I~IV systems correspond to the recurrent DNN with the recurrent units at 2nd, 3rd, 4th and 5th hidden layer from the input layer using the introduced truncated BPTT as described in Section 3.3.2.

Systems	Conditions						Avg.
	9dB	6dB	3dB	0dB	-3dB	-6dB	
DNN IV	16.89	20.29	22.83	30.36	39.49	49.47	29.89
RDNN	17.26	20.29	22.83	30.21	38.67	48.98	29.71
RDNN I	15.92	18.59	21.41	28.28	**35.81**	47.23	27.87
RDNN II	15.95	18.49	**20.87**	**27.89**	36.65	**46.35**	**27.70**
RDNN III	16.22	18.49	21.24	28.04	36.26	46.46	27.79
RDNN IV	**15.84**	**18.16**	21.03	28.21	36.93	47.17	27.89

Table 9: WERs (%) of best DNN-HMM system and recurrent DNN-HMM systems trained on CHiME challenge multi-condition data with available stereo data.

Systems	Conditions						Avg.
	9dB	6dB	3dB	0dB	-3dB	-6dB	
DNN V	14.27	16.44	19.39	24.68	31.65	42.05	24.75
RDNN V	**13.60**	**14.96**	**17.92**	**22.98**	**29.07**	**38.11**	**22.77**

7.3% relative improvement over our best DNN system. Furthermore, we observe no significant performance difference between different setups of recurrent DNN, but the system seems working best with the architecture where the recurrent layer is located at the middle of the DNN. Finally we conduct the experiments on the dataset with the assumption that the stereo data is available. We train the similar GMM-HMM system on the clean speech data, and then using the alignments on clean data as the label to train the DNN and recurrent DNN with similar setup as described earlier. The experimental results are list in Table 9, as can be seen that recurrent DNN also obtain consistent and significant performance gain over the DNN system.

3.4.2 Experiments on Aurora-4

Aurora-4 is also a medium vocabulary task based on the WSJ0 corpus. The training set contains both 8kHz and 16 kHz multi-condition 7137 utterances from 83 speakers. One

Table 10: WERs (%) of baseline GMM-HMM, and DNN-HMM systems on Aurora-4 data, DNN I~III systems correspond to the iteratively retrained DNNs with the new alignments.

Systems	Conditions				Avg.
	Clean	Noise	Channel	Channel+Noise	
GMM	8.28	13.83	17.84	29.24	20.32
DNN I	3.51	8.15	10.69	22.59	14.19
DNN II	3.34	7.48	10.09	21.76	13.49
DNN III	**3.24**	**7.44**	**10.07**	**21.43**	**13.33**

half of the utterances were recorded by the primary closed microphone and the other half were recorded using one of secondary open microphones. Among the whole training set, there are six different types of noise backgrounds, street traffic, train station, car, babble, restaurant, airport at $10 \sim 20$ dB SNR. The evaluation set is noisy and reverberated version of WSJ0 5K NOV-92 which consists of 4620 utterances in 14 conditions (330×14) and can be grouped into 4 subsets: clean, noisy, clean with channel distortion, noisy with channel distortion.

We first build the baseline GMM-HMM system for the tasks, 2026 distinct tied-state GMMs are trained with MFCC features coupled with their linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT) on the 7137 multi-condition speech utterances. For the DNN-HMM systems, the setup is similar to the one in previous experiment: we first generatively pretrain the DNN using RBMs, and stack them to initialize the DNN with 7 2048-dim hidden layers; With the alignments from GMM-HMM systems, we trained the DNN I system using 40-dim log Mel filter-bank features. Then we further do the realignments using trained DNN I system and train the second DNN system with the new alignments. Then this process is repeated until there are no further significant improvements. The experimental results are shown in Table 10. The best DNN III system achieves 13.33% average WER. Following the same setup as described in the experiment on CHiME challenge data, we build two recurrent DNN systems on top of DNN III system, namely RDNN I and II systems which correspond to the recurrent DNN with the recurrent

Table 11: WERs (%) of best DNN-HMM system and two recurrent DNN-HMM systems trained on Aurora-4 multi-condition data: RDNN I, II systems correspond to the recurrent DNN with the recurrent units at the 3rd and 4th hidden layer from the input layer.

Systems	Conditions				Avg.
	Clean	Noise	Channel	Channel+Noise	
DNN III	3.34	7.44	10.07	21.43	13.33
RDNN I	3.27	7.30	9.15	20.67	12.88
RDNN II	**3.06**	**7.26**	**9.10**	**20.44**	**12.74**

units at the 3rd and 4th hidden layer from the input layer. As shown in Table 11, both recurrent DNN system outperform the DNN III system in all conditions. The RDNN II system achieves 0.59% absolute improvements over our best DNN system.

3.5 Summary

In this chapter, we propose recurrent DNNs for robust acoustic modeling. A new BPTT algorithm is introduced to make the minibatch SGD on the proposed recurrent DNNs more efficient and effective. We evaluate the proposed recurrent DNN architecture under the hybrid setup on both 2nd CHiME challenge (track 2) and Aurora-4 tasks. The experimental results on the CHiME challenge data show that we can obtain consistent 7% relative WER improvements over DNN systems, achieving the state-of-the-art performance without front-end preprocessing, speaker adaptive training or multiple decoding passes. On the Aurora-4, the proposed system obtains 4% relative WER improvement over a strong DNN baseline system.

CHAPTER 4

SINGLE-CHANNEL MIXED SPEECH RECOGNITION USING DEEP NEURAL NETWORKS

4.1 Introduction

While significant progress has been made in improving the noise robustness of speech recognition systems, recognizing speech in the presence of a competing talker remains one of the most challenging unsolved problems in the field. To study the specific case of single-microphone speech recognition in the presence of competing talker, a monaural speech separation and recognition challenge [89] was issued in 2006. It enabled researchers to apply a variety of techniques on the same task and make comparisons between them. Several types of solutions were proposed. Model based approaches [1, 2, 90] use factorial GMM-HMM [91] to model the interaction between target and competing speech signals and their temporal dynamics, then the joint inference or decoding determined the two most likely speech signals or spoken sentences given the observed speech mixture. In computational auditory scene analysis (CASA) and missing feature approaches [92, 93, 94], certain segmentation rules operate on low-level features to estimate a time-frequency mask that isolates the signal components that belong to the each speaker. This mask is used either to reconstruct the signal or directly inform the decoding process. Some other approaches including [95] and [96] utilize the non-negative matrix factorization (NMF) for the separation and pitch-based enhancement. Among all the submissions to the challenge, the IBM superhuman system [1] performed the best and even exceeded what human listeners could do on the challenge task (see Table 12). Their system consists of three main components: a speaker recognizer, a separation system, and a speech recognizer. The separation system requires as input the speaker identities and signal gains that are output from the speaker recognition system. In practice, it is usually necessary to enumerate several of the most probable speaker combinations and run the whole system multiple times. This may be impractical when the

Table 12: Overall keywords WERs of three systems/methods on the 2006 challenge task. IBM superhuman: Hershey et al. [1] ; Human: human listeners; Next best: the system by Viranen [2].

System/Method	IBM superhuman	Human	Next best
WER	21.6%	22.3%	34.2%

number of speakers is large. The separation system uses factorial GMM-HMM generative models with 256 Gaussians to model the acoustic space for each speaker. While this was sufficient for the small vocabulary in the challenge task, it is a very primitive model for a large vocabulary task. However, with a larger number of Gaussians, performing inference on the factorial GMM-HMM becomes computationally impractical. Moreover, the system assumes the availability of speaker-dependent training data and a closed set of speakers between training and test.

Recently, acoustic models based on deep neural networks (DNNs) [31] have shown great successes on both LVCSR [78] and robust ASR tasks [80] . However, few, if any, previous work has explored how DNNs could be used in the multi-talker speech recognition scenario. High-resolution features are typically favored by speech separation system, while the fact that a conventional GMM-HMM ASR system is incapable of compactly modeling the high-resolution features usually forces researchers to perform speech separation and recognition separately. However, DNN-based systems have been shown to work significantly better on spectral-domain features than cepstral-domain features [97], and have shown outstanding robustness to speaker variation and environment distortions [79]. In this chapter, we aim to build a unified DNN-based system, which can simultaneously separate and recognize two-talker speech in a manner that is more likely to scale up to a larger task. We propose several methods for co-channel speech recognition that combine multi-style training with different objective functions defined specifically for the multi-tasker task. The phonetic probabilities output by the DNNs will then be decoded by a WFST-based decoder modified to operate on multi-talker speech. Experiments on the 2006 speech separation and recognition challenge data demonstrate the proposed DNN based

system has remarkable noise robustness to the interference of competing talker. The best setup of our systems achieves 18.8% overall WER, which is 2.8% absolute improvement over the state-of-the-art IBM system with less complexity and fewer assumptions.

The remainder of this chapter is organized as follows. We first review DNN based approaches to robust speech recognition in 4.2. In Section 4.3, we describe our multi-style DNN training and the different multi-talker objective functions used to train the networks. The WFST-based joint decoder is introduced in Section 4.4. We report experimental results in Section 4.5 and summarize our work in Section 4.6.

4.2 DNN-based Approaches to Noise Robust ASR

DNN-based approaches to noise robustness for ASR generally can be categorized into two types: feature space enhancement/compensation or model space adaptation. In the feature enhancement methods, DNNs or recurrent neural networks (RNNs) are treated as front-end denoisers [82, 83] which try to capture the mapping relations between the noise and clean speech. In these methods, DNNs are usually trained on clean and noise stereo pairs of observations based on the minimum mean squared error (MMSE) criterion [79]. Similar to LDA [71] and fMLLR [72], the mismatched acoustical conditions between training and testing in the feature space can also be compensated by learning feature transformations as in the linear input network (LIN) [98] or feature discriminative linear regression (fLDR) [99]. Another common use of DNNs in the feature space for noise robustness is known as Tandem systems. The probabilistic Tandem system [28] concatenates the neural network output posteriors with original MFCCs features and feeds this augmented features to the conventional GMM-HMM system. The bottleneck Tandem system [29] has become popular in recent LVCSR systems [100, 101], where a DNN with a bottleneck hidden layer in the middle is trained and the linear output of the bottleneck layer is taken as output instead of the posteriors. RNNs have also been used in the Tandem setups for enhancing noise robustness in [81]. The advantage of all these feature space approaches is the flexibility in

the back-end selection of the system, where all kinds of back-end techniques can be applied directly.

In the DNN-based model space adaptation methods, DNNs are used in the DNN-HMM hybrid setup to generate the acoustic score for each HMM state emission. In the conventional GMM-HMM LVCSR system, the state emission log-likelihood of the observation feature vector x_t for certain tied state or senone s_j of HMMs is generated using,

$$\log p(x_t|s_j) = \log \sum_{m=1}^{M} \pi_{jm} \mathcal{N}_{jm}(x_t|s_j), \tag{69}$$

where M is the number of Gaussian mixtures in the GMM for state j and π_{jm} is the mixing weight. While the outputs from DNNs represent the state posteriors $p(s_j|x_t)$, a DNN-HMM hybrid system [31] uses pseudo log-likelihood as the state emissions,

$$\log p(x_t|s_j) \propto \log p(s_j|x_t) - \log p(s_j), \tag{70}$$

where the state priors $\log p(s_j)$ can be estimated using the state alignments on the training speech data. The input features vectors x_t to the first layer of DNNs usually use a context of l frames [31], *e.g.*, $l = 9$ or $l = 11$. The most straightforward way of DNN-based model adaptation methods is multi-style training strategy [73]: first creating a number of artificial acoustical environments by corrupting the clean database with noise samples of various levels and types; then training DNNs with all those created multi-condition waveforms. Recently, multiple DNN model training methods, e.g, noise-aware training and dropout, have been introduced in [80] to lead more accurate senone prediction under various noise conditions for DNN-HMM hybrid system. In [102], the recurrent architecture is introduced into the DNN-HMM hybrid system and the authors can achieve state-of-the-art performances on both the 2nd CHiME challenge (track 2) [85] and Aurora-4 tasks without front-end preprocessing, speaker adaptive training or multiple decoding passes. Recently long short-term memory (LSTM) recurrent architectures are also introduced in the hybrid system [103] [104] for more robust acoustic modeling.

4.3 DNN Multi-style Training with Mixed Speech

Although a DNN-based acoustic model has proven to be more robust to environmental perturbations, it was also shown in [79] that the robustness holds well only for the input features with modest distortions beyond what was observed in the training data. When there exist severe distortions between training and test samples, it is essential for DNNs to see examples of representative variations during training in order to generalize to the severely corrupted test samples. Since that we are dealing with a challenging task where the speech signal from the target speaker is mixed with a competing one, a DNN-based model will generalize poorly if it is trained only on single-speaker speech, as will be shown in Section 4.5. As mentioned in 4.2, one way to circumvent this issue is using a multi-style training strategy [73] in which training data is synthesized to be representative of the speech expected to be observed at test time. In our case, this means corrupting the clean single-talker speech database with samples of competing speech from other talkers at various levels and then training the DNNs with these created multi-condition waveforms. In this section, we describe how this multi-condition data can be used to create networks that can separate multi-talker speech.

4.3.1 High and Low Energy Signal Models

In each mixed-speech utterance, we assume that one signal is the target speech and one is the interference. The labeling is somewhat arbitrary as the system will decode both signals. The first approach assumes that one signal has higher average energy than the other. Under this assumption, we can identify the target speech as either the higher energy signal (positive SNR) or the lower energy signal (negative SNR). Thus in our first system, two DNNs are used: given a mixed-speech input, one network is trained to recognize the higher energy speech signal while the other one is trained to recognize the low energy speech signal which we will refer to as the high and low energy models respectively. Suppose we are given a clean training dataset $\mathcal{X}$, we first perform energy normalization so that each speech utterance in the data set has the same power level. To simulate the acoustical environments

where the target speech signal has higher average energy or lower average energy, we randomly choose another signal from the training set, scale its amplitude appropriately and mix it with the target speech: for the multi-condition dataset used to train the high energy signal models, we need to decrease the amplitude of those speech waveforms mixed with the target speech to various levels; for the multi-condition dataset used to train the low energy signal models, we need to increase the amplitude level accordingly. Denote by $\mathcal{X}_H$,$\mathcal{X}_L$ the two multi-condition datasets created as described, for the high energy target speaker, we train the DNN models with the loss function,

$$\mathcal{L}_{\text{CE}}(\theta) = -\sum_{x_t \in \mathcal{X}_H} \log p(s_j^{\text{H}} | x_t; \theta), \tag{71}$$

where s_j^{H} is the reference senone label at t^{th} frame. Note that the reference senone labels come from the alignments on the uncorrupted data. This was critical to obtaining good performance in our experiments. Similarly, the DNN models for the low energy target speaker can be trained on the dataset $\mathcal{X}_L$.

4.3.2 High and Low Energy Signal Denoisers

As mentioned in Section 4.2, in the feature enhancement approaches to robust ASR, DNNs are treated as front-end denoisers. With the same two created dataset $\mathcal{X}_L$ and $\mathcal{X}_H$, the front-end deep denoiser can also be trained based on minimum square error (MSE) loss function,

$$\mathcal{L}_{\text{MSE}}(\theta) = \sum_{x_t \in \mathcal{X}_H} |\hat{y}(x_t; \theta) - y_t|^2, \quad y_t \in \mathcal{X}, \tag{72}$$

where $y_t \in \mathcal{X}$ is the corresponding clean speech features, *i.e.*, the features generated on the original uncorrupted target speech, and $\hat{y}(x_t; \theta)$ is the estimation of the uncorrupted inputs using the deep denoiser. Similarly, the denoiser for the low energy target speaker can be trained on the dataset $\mathcal{X}_L$.

4.3.3 High and Low Pitch Signal Models

One potential issue with the above training strategy based on high and low energy speech signals is that the trained models may perform poorly when mixed signals have similar

average energy levels, *i.e.*, near 0dB SNR. The reason is that the problem is ill-defined in this region since one cannot reliably label one signal as the higher or lower energy signal. Since it is far less likely that the two speakers will speak with the same pitch, we propose another approach where DNNs are trained to recognize the speech with the higher or lower pitch which we will refer to as the high and low pitch signal models. In this case, we only need to create a single training set $\mathcal{X}_P$ from original clean dataset $\mathcal{X}$ by randomly choosing an interfering speech signal and mixing it with the target speech signal. The training also requires a pitch estimate for both the target and interfering speech signals which will be used to select appropriate labels for DNN training, *i.e.*, the senone labels always come from the alignments on the speech utterances with the higher average pitch when the high pitch signal models are being trained. The loss function for training the DNN for the high pitch speech signals is thus,

$$\mathcal{L}_{\text{CE}}(\theta) = -\sum_{x_t \in \mathcal{X}_P} \log p(s_j^{\text{HP}}|x_t;\theta), \tag{73}$$

where s_j^{HP} is the reference senone label obtained from the alignments on the speech signal with the higher average pitch. Similarly, a DNN for the lower pitch speech signals can be trained with the senone alignments of the speech signal with the lower average pitch.

4.3.4 Instantaneous High and Low Energy Signal Models

Still, the limitations of both high/low energy and pitch signal models lie in their weakness on dealing with the scenarios when two speakers talk at similar average energy or pitch level. This motivates us to train the models based on the characteristics of each individual frame rather than the whole utterance which we will refer to as instantaneous high and low energy models, *i.e.*, we can train the DNNs based on the instantaneous energy in each frame rather than the average energy of the utterance. Even an utterance with an average energy of 0 dB will have non-zero instantaneous SNR values in each frame, this means there is no ambiguity in the labeling. We only need to create one training set $\mathcal{X}_I$ by mixing speech signals and computing the instantaneous frame energies in the target and interfering signal.

The loss function for the instantaneous high energy signal is given by,

$$\mathcal{L}_{\text{CE}}(\theta) = -\sum_{x_t \in \mathcal{X}_I} \log p(s_j^{\text{IH}} | x_t; \theta), \tag{74}$$

where s_j^{IH} corresponds to the senone label from the signal source which contains higher energy at frame t,

$$s_j^{\text{IH}} = \begin{cases} s_j^1 & \text{Energy}(x_t^1) > \text{Energy}(x_t^2) \\ s_j^2 & \text{Energy}(x_t^1) < \text{Energy}(x_t^2) \end{cases}. \tag{75}$$

Similarly, the instantaneous low energy signal models can be trained with the senone labels assigned using (75) but with reversed signs in the inequality conditions. However, in this case, since we are using a frame-based energy rather than an utterance-based energy as the criterion for separation, there is uncertainty as to which output corresponds to the target or interferer from frame to frame during the decoding stage. For example, the target speaker can have higher energy in one frame and lower energy in the next frame. We will address this in the decoder described in the next section.

4.4 Joint Decoding with DNN models

For the DNNs based on instantaneous energy, we need to determine which of the two DNN outputs belongs to which speaker at each frame. To do so, we introduce a WFST based joint decoder that can take the posterior probability estimates from the instantaneous high-energy and low-energy DNNs to jointly find best two state sequences, one for each speaker.

The standard recipe for creating the decoding graph in the WFST framework [23] can be written as,

$$\text{HCLG} = \min(\det(H \circ C \circ L \circ G)), \tag{76}$$

where H, C, L and G represent the HMM structure, phonetic context-dependency, lexicon and grammar respectively, and $\circ$ is WFST composition. The input labels of the HCLG are the identifiers of context-dependent HMM states (senone labels), and the output labels represent words. The HCLG graph has encoded all the information needed to decode an

utterance, *i.e.*, HMM topologies and transitions probabilities, lexicon and pronunciation model scores and languages model scores except the acoustic scores which only can be evaluated when the speech frames to be decoded become available. Suppose now we want to decode an utterance with T frames, the acoustic scores information can be encoded into a $(T + 1)$-state WFSA U where both input and output labels of each arc between states are HMM transition identifiers and the cost is the corresponding acoustic score, *i.e.*, $p(x_t|s_j)$, under Log or Tropical semiring. Decoding the utterance is essentially finding the best path in the $U \circ HCLG$ graph. But the graph $U \circ HCLG$ is never explicitly constructed as it is extremely large without pruning. Instead, finding the best path is done via token passing on the HCLG graph: At frame t, each active token is associated one state in the HCLG graph; Then we consume one more speech frame by passing all the active tokens through the arcs and accumulate the corresponding acoustic scores; At frame $(t + 1)$, all the tokens fall outside the beam-width are cut off. After the token passing is done, we can then trace back the best sequence from the information stored in the dynamic programming table.

4.4.1 Joint Token Passing on the HCLG Graphs

The task now is to find best two state sequence in the 2-D joint state space whose size will be the square of the size for each individual speaker's state space. The key part of our proposed decoding algorithm is joint token passing on the two HCLG decoding graphs in conjunction with the acoustic scores accumulations using instantaneous high and low energy DNN models. The main difference in token passing between our joint decoding and conventional decoding is that now each token is associated with two states rather than one in the decoding graph. Denote by θ^{H} and θ^{L} instantaneous high and low energy signal DNN models trained as described in Section 4.3.4. The joint decoder is to find best two state sequence such that the sum of each joint state-sequence log-likelihood is maximized,

$$(\mathbf{s}^{1*}, \mathbf{s}^{2*}) = \underset{(\mathbf{s}^1,\mathbf{s}^2)\in\{\mathbf{s}^1\times\mathbf{s}^2\}}{\operatorname{argmax}} \Big\{ p(x_{1:T}|\mathbf{s}^1; \theta^{\mathrm{H}}, \theta^{\mathrm{L}}) p(\mathbf{s}^1) \cdot p(x_{1:T}|\mathbf{s}^2; \theta^{\mathrm{H}}, \theta^{\mathrm{L}}) p(\mathbf{s}^2) \Big\}. \tag{77}$$

Figure 12 shows a toy example to illustrate the joint token passing process: suppose the token for the first speaker is at state 1, and the token associated with the second speaker is at state 2. For the outgoing arcs with non-ϵ input labels (those arcs that consume acoustic frames), the expanded arcs we will pass the tokens through are the Cartesian product between the two outgoing arc sets. The graph cost of each expanded arc will be the semiring multiplication of the two. The acoustic cost of each expanded arc is computed using the senone hypotheses from the two trained DNNs for the instantaneous high and low energy. Because we need to consider both cases where either one of the two sources has the higher energy, the acoustic cost is given by the combination with higher likelihood,

$$C = \max\{p(x_t|s^1;\theta^{\mathrm{H}}) \cdot p(x_t|s^2;\theta^{\mathrm{L}}), \\ p(x_t|s^1;\theta^{\mathrm{L}}) \cdot p(x_t|s^2;\theta^{\mathrm{H}})\}. \tag{78}$$

With the equation above, we can also tell which speaker has higher energy in the corresponding signal at certain frame t along this search path. For the arcs with ϵ input labels, the expansion process is bit tricky. As the ϵ arcs are not consuming acoustic frames, to guarantee the synchronization of the tokens on two decoding graphs, a new joint state for

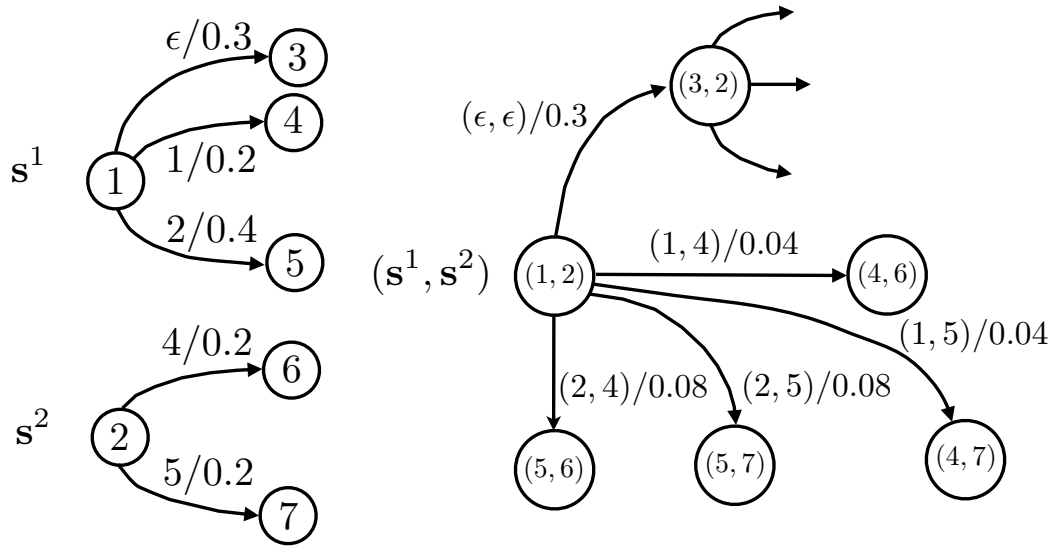

Figure 12: A toy example illustrating the joint token passing on the two WFST graph: $\mathbf{s}^1$, $\mathbf{s}^2$ denote state space corresponds to one of two speakers; $(\mathbf{s}^1, \mathbf{s}^2)$ represent the joint state space.

current frame has to be created (see the state $(3, 2)$ in the Fig.12). And for each newly created joint states, we repeat the same joint token passing process until all the tokens are processed.

4.4.2 Penalties on Energy Switching

One potential issue of our joint decoder is that we allow free energy switching frame by frame while decoding the whole utterance. Yet, we know that in practice, the energy switching should not typically occur too frequently. This issue can be overcome by introduce a constant penalty in certain searching path when the louder signal has changed from last frame so that the state sequences with both relatively high likelihood and low energy switch frequency will survive in the end. Recall that when we compute the acoustic cost during joint decoding, at the same time we know which speaker speaks louder at certain frame along this search path, we keep this information when doing token passing and beam searching. If the louder signal has changed from last frame, we add a constant cost to this searching path.

Alternatively, we can estimate the probability that a certain frame is the energy switching point and let the value of the penalty adaptively changed with it. Since we created the training set by mixing the speech signals, the energy of each original speech frame is available. We can use it to train a DNN to predict whether the energy switch point occurs at certain frame. If we let θ^S represent the models we trained to detect the energy switching point, the adaptive penalty on energy switching is given by,

$$\mathcal{P} = -\alpha \cdot \log p(y_t|x_t; \theta^S). \tag{79}$$

4.5 Experiments

In this section, we report our experimental results with all proposed systems on the 2006 monaural speech separation and recognition challenge data [89].

4.5.1 The Challenge Task and Scoring Procedure

The main task of 2006 monaural speech separation and recognition challenge is to recognize the keywords (numbers and letters) from the speech of a target speaker in the presence of another competing speaker using a single microphone. The speech data of the challenge task is drawn from GRID corpus [105]. The training set contains 17,000 clean speech utterances from 34 difference speakers (500 utterances for each speaker). The evaluation set includes 4,200 mixed speech utterances in 7 conditions, clean, 6dB, 3dB, 0dB, -3dB, -6dB, -9dB target-to-mask ratio (TMR) and the development set contains 1,800 mixed speech utterances in 6 conditions (no clean condition). As shown in Fig 13, the fixed grammar contains six parts: command, color, preposition, letter (with W excluded), number, and adverb, *e.g.*, "place white at L 3 now". During the test phase, the speaker who utters the color 'white' is treated as the target speaker. The evaluation metric is the WER on letters and numbers spoken by the target speaker. Note that the WER on all words will be much lower, and unless otherwise specified, all reported WERs in the following experiments are the ones evaluated only on letters and numbers.

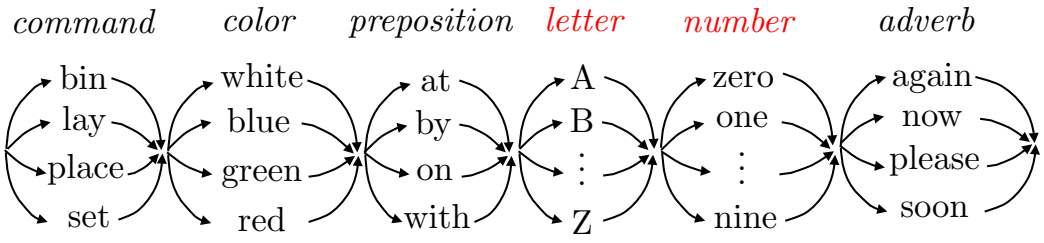

Figure 13: The grammar of 2006 monaural speech separation and recognition challenge contains six parts: command, color, preposition, letter (with W excluded), number, and adverb; The evaluation metric is the WER on letters and numbers spoken by the target speaker.

4.5.2 Baseline System

The baseline system is built using a DNN trained on the original training set consisting of 17,000 clean speech utterances. We first train a GMM-HMM system using 39-dimension MFCCs features with 271 distinct senones. Then we use 64 dimension log mel-filterbank

Table 13: WERs (%) of baseline GMM-HMM and DNN-HMM systems: both systems are trained on the clean training data.

Systems	Conditions						
	Clean	6dB	3dB	0dB	-3dB	-6dB	-9dB
GMM	4.0	38.5	54.7	70.5	82.3	89.3	94.2
DNN	0.7	32.5	48.8	66.3	78.4	86.3	91.8

as features and context window of 9 frames to train the DNN. The DNN has 7 hidden layers with 1024 hidden units at each layer and the 271-dimensional softmax output layer, corresponding to the senones of the GMM-HMM system. The following training scheme will be used through all the DNN experiments: the parameter initialization is done using layer by layer using generative pre-training [106] following by discriminative pre-training [99]. Then the network is discriminatively trained using backpropagation. The mini-batch size is set to 256 and the initial learning rate is set to 0.008. After each training epoch, we validate the frame accuracy on the development set, if the improvement is less than 0.5%, we shrink the learning rate by the factor of 0.5. The training process is stopped after the frame accuracy improvement is less than 0.1%. The WERs of the baseline GMM-HMM and DNN-HMM system are shown in Table 13. As can be seen, the DNN-HMM system trained only on clean data performs poorly in all SNR conditions except the clean condition, confirming the necessity of DNN multi-style training.

4.5.3 Multi-style Trained DNN Systems

To investigate the use of multi-style training for the high and low energy signal models, we generated two mixed-speech training datasets:

I. The high energy training set, which we refer to as Set I, was created as follows: for each clean utterance, we randomly choose three other utterances and mixed them with the target clean utterance under 4 conditions, clean, 6dB, 3dB, 0dB. (17,000 $\times$ 12).

II. The low energy training set, referred to as Set II, was created in a similar manner but the mixing was done under 5 conditions, clean, and TMRs of 0dB, -3dB, -6dB, -9dB.

Table 14: WERs (%) of the DNN systems for high and low energy signals; DNN I: multi-style trained DNN for the high energy signals; DNN II: multi-style trained DNN for the low energy signals; DNN I+II: the combined system of I and II using the rule that the target speaker is the one who speaks color 'white'.

Systems	Conditions						AVG
	6dB	3dB	0dB	-3dB	-6dB	-9dB	
DNN	32.5	48.8	66.3	78.4	86.3	91.8	67.4
DNN I	**4.5**	**16.8**	56.8	-	-	-	-
DNN II	-	-	52.6	33.6	**18.4**	**17.4**	-
IBM [1]	15.4	17.8	**22.7**	**20.8**	22.1	30.9	**21.6**
DNN I+II	**4.5**	16.9	49.8	39.8	21.7	19.6	25.4

(17,000 $\times$ 15).

Then we use these two training sets to train two DNN models, DNN I and II, for high and low energy signals respectively, and listed the results in Table 14. From the table, we can see the results are surprisingly good, especially in the cases where two mixing signals have large energy level difference, *i.e.*, 6dB, -6dB, -9dB. By combining the results from DNN I and II systems using the rule that the target speaker always utters the color white, the combined DNN I+II system achieves 25.4% WER compared to 67.4% which obtained with the DNN trained only on clean data.

For the high and low pitch signals models, we first estimate the pitch for each speaker from the clean training set. Then we combine the Train Set I and Train Set II to form Set III (17,000 $\times$ 24) to train two DNNs for high and low pitch signals respectively. When training the DNNs for the high pitch signals, we assign the label from the alignments on clean speech utterances corresponding to the high pitch talker; When training the DNNs for the low pitch signals, we assign the label from the alignments corresponding to the low pitch talker. With the two trained DNN models, we do the decoding independently as before and combine the decoding results using the rules that the target speaker always utters the color white. We list the WERs in Table 15. As can be seen, the system with the high and low pitch signal models performs better than the one with the high and low energy models in the 0dB case, but worse in the other cases.

Table 15: WERs (%) of the DNN systems for high and low pitch signals; DNN III: multi-style trained DNN for high and pitch signals.

Systems	Conditions						AVG
	6dB	3dB	0dB	-3dB	-6dB	-9dB	
DNN I+II	**4.5**	**16.9**	49.8	39.8	**21.7**	**19.6**	**25.4**
DNN III	14.5	22.1	**30.8**	41.9	52.8	59.6	36.9

4.5.4 Multi-style Trained Front-end Denoisers

Then we experimented with the multi-style trained deep denoiser. With the same training set I, we train a DNN as a front-end denoiser as described in Section 4.3.2. Note that the top softmax layer is removed in the denoiser and the features are mean and variance normalized before fed into the DNNs. With trained deep denoisers, we try two different setups: the first one we directly feed denoised features to the DNN trained on the clean data; in the second setup, we retrained another DNN on the denoised data and conduct the experiments. We list the results of both setups in the Table 16. From the above experiments, there are two noteworthy points. First, the system with the DNN trained to predict senone labels seems slightly better than the one with a trained deep denoiser followed by another retrained DNN. This implies that DNN is capable learning robust representations automatically, there may be no need to extract hand-crafted features in the front-end. The combined system DNN I+II is still not good as the state-of-the-art IBM superhuman system. The main reason is that the system performs very poorly in the cases where two mixing signals have very close energy level, *i.e.*, 0dB, -3dB. This coincides with our concerns discussed earlier. Specifically, the multi-style training strategy for the high and low energy signals has the potential issue of assigning conflicting labels during training.

To take an insightful look at the denoised features, we select features of one test sample speech utterance under clean, 6db, 3db, 0db conditions, feed them into the trained deep denoisers for the high energy signals and plot the original input features and denoised output features in Fig 14-17. Each figure shows pairs of input and output filter-bank features under clean, 6db, 3db, 0db conditions respectively (note that the input features are mean

Table 16: WERs (%) of deep denoisers for high energy signals; Denoiser I + DNN: the system where we feed denoised features to the DNN train on clean data. Denoiser I + DNN (retrained): the system where we retrained the DNN on the denoised feature.

Systems	Conditions		
	6dB	3dB	0dB
Denoiser I + DNN	16.8	32.2	65.9
Denoiser I + DNN (retrained)	6.3	17.3	**56.3**
DNN I	**4.5**	**16.8**	56.8

and variance normalized.). The squared errors averaged over all time-frequency bins under four conditions are 0.1062, 0.1896, 0.2768 and 0.5174. From the figures, we can also tell that the denoiser works fairly good in 6db and 3db conditions where the energy in some time-frequency bins belonging to the interference speaker can be removed. However, the residuals under 0db becomes very severe which illustrates why the high and low energy signal models perform poorly in the cases where two mixing signals have similar energy levels.

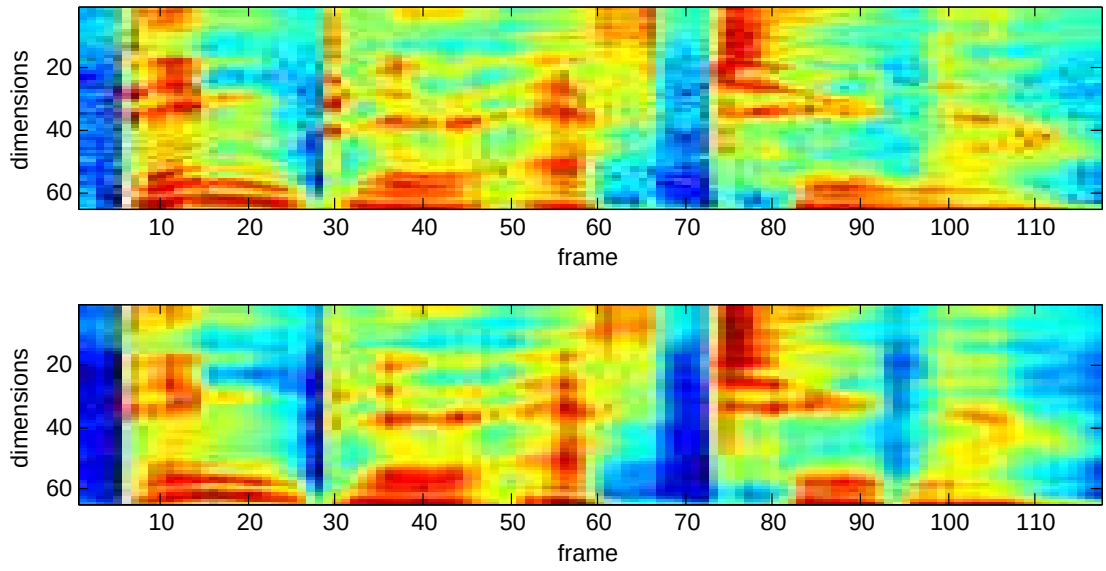

Figure 14: Upper: the mean variance normalized mel-scale filter-bank features of the sample utterance under the clean condition; Bottom: the reconstructed filter-bank features from the high energy denoisers, averaged squared error is 0.1062.

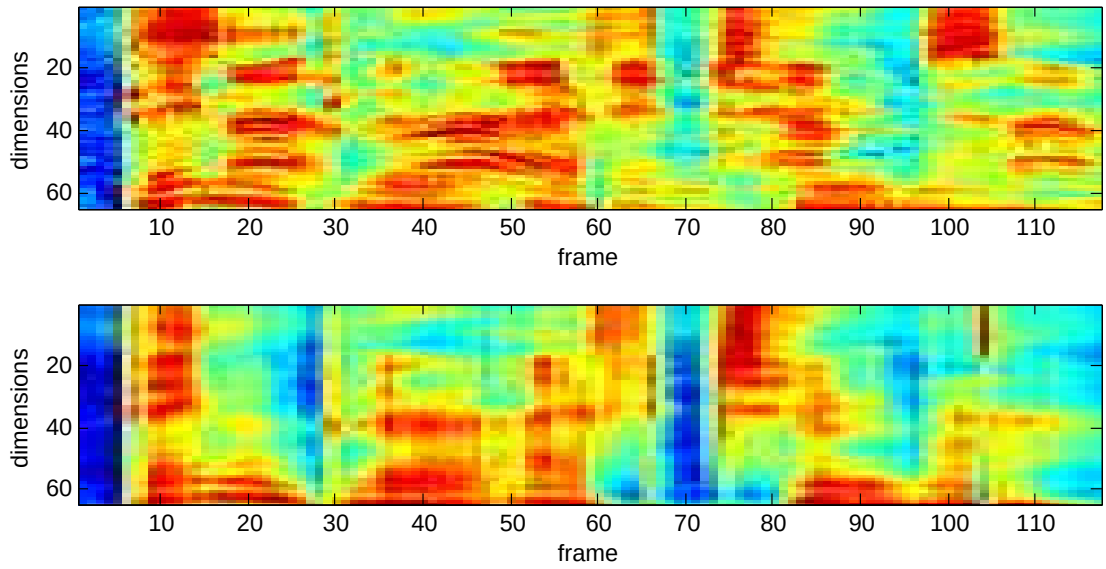

Figure 15: Upper: the mean variance normalized mel-scale filter-bank features of the sample utterance under the 6dB condition; Bottom: the reconstructed filter-bank features from the high energy denoisers. Some obvious interference time frequency bins have been suppressed or removed, averaged squared error is 0.1896.

4.5.5 DNN System with Joint Decoder

Finally, we use training set III to train two DNN models for instantaneous high and low energy signals as described in Section 4.3.4. With these two trained models, we perform a joint decoding as described in Section 4.4. The results of this Joint Decoder approach are shown in Table 17. The last two systems correspond to the cases where we introduce the energy switching penalties. The Joint Decoder I is the system with the constant energy switching penalty and Joint Decoder II is the system with adaptive switching penalty. To get the value of the energy switching penalties as defined in (79), we trained a DNN to estimate an energy switching probability for each frame.

From Table 17, we can see that the DNN I+II system performs well in the cases where two mixing speech signals have large energy level difference, *i.e.*, 6dB, -6dB, -9dB, while the Joint Decoder II system performs well in the cases where two mixing signals have similar energy level. This motivates us to do the system combination according to the energy level differences between the two signals. Note that if the SNRs of the input speech signals are available, system combinations can be directly done via selecting either the

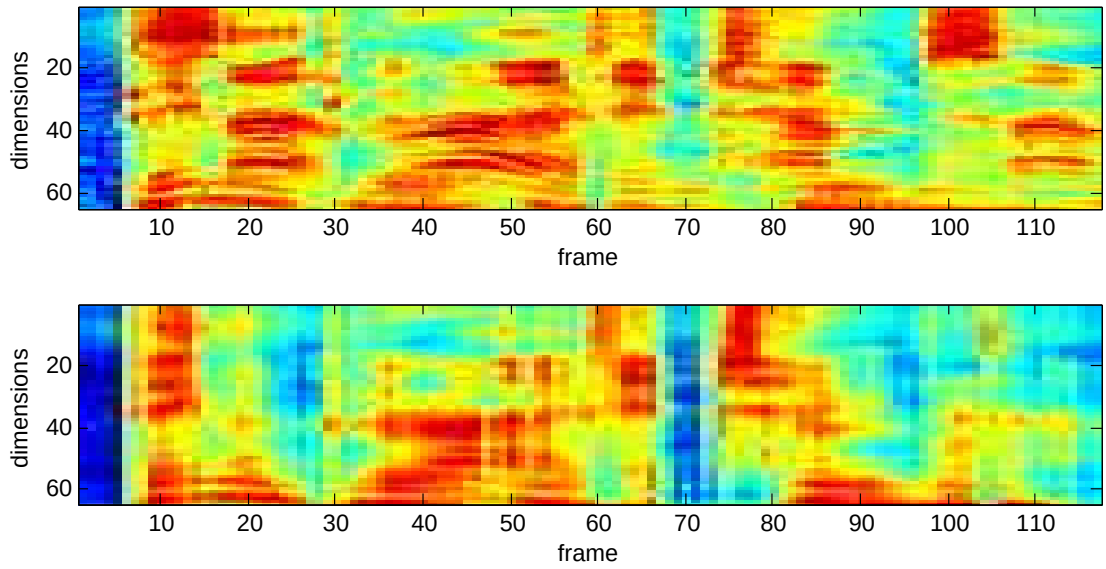

Figure 16: Upper: the mean variance normalized mel-scale filter-bank features of the sample utterance under the 3dB condition; Bottom: the reconstructed filter-bank features from the high energy denoisers, averaged squared error is 0.2768.

multi-trained DNN I+II or the joint decoder system. We list this oracle WERs with the assumptions that the SNR level of each testing utterance is known in Table 18. If the SNR levels are not available, we need to introduce the mechanisms to determine which system works better. In Section 4.5.6 and 4.5.7, we will present two ways to do system combinations based on the front-end denoisers and lattice confidence scores.

Table 17: WERs (%) of the DNN systems with the joint decoders; Joint Decoder: the joint decoder system without the energy switching penalties; Joint Decoder II: the joint decoder system with the constant energy switching penalties inserted; Joint Decoder III; the joint decoder system with the adaptive switching penalties.

Systems	Conditions						AVG
	6dB	3dB	0dB	-3dB	-6dB	-9dB	
DNN	32.5	48.8	66.3	78.4	86.3	91.8	67.4
IBM [1]	15.4	17.8	22.7	20.8	22.1	30.9	21.6
DNN I+II	**4.5**	**16.9**	49.8	39.8	**21.7**	**19.6**	25.4
Joint Decoder	18.3	19.8	**19.3**	21.3	23.2	27.4	21.5
Joint Decoder I	16.1	18.7	20.5	19.6	23.6	26.8	20.9
Joint Decoder II	16.5	17.1	19.9	**18.8**	22.5	25.3	**20.0**

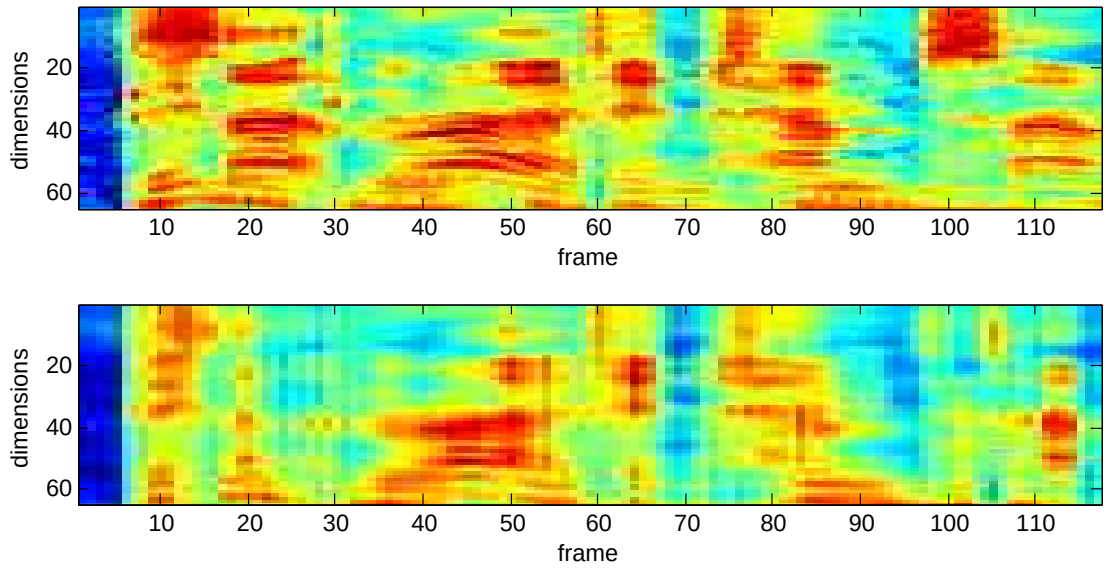

Figure 17: Upper: the mean variance normalized mel-scale filter-bank features of the sample utterance under the 0dB condition; Bottom: the reconstructed filter-bank features from the high energy denoisers, the reconstruction residuals become severe, averaged squared error is 0.5174.

4.5.6 System Combination Using Deep Denoisers

One way to combine the two systems is based on the energy levels difference of two mixing speech signals. To get energy level difference between two mixing signals, we can use the front-end deep denoisers for the high and low energy signals. The mixed signal is input to the two deep denoisers and the two resultant output signals will be used to estimate the high and low energy signals. Using these separated signals, we can calculate their energy ratio to approximate the energy difference of two original signals. Note that since both the input and denoised features of the front-end deep denoisers are mean and variance normalized, we need to transform the denoised features back to the unnormalized ones when calculating the energy level difference. We first tune and obtain a optimal threshold for the energy ratio on the development set, and use it for the system combination, *i.e.*, if the energy ratio of two separated signals from the denoisers is higher than the threshold, we use system DNN I+II to decode the test utterance, otherwise the system Joint Decoder II will be used. The results are listed in Table 18.

Table 18: WERs (%) of the combined systems using DNN I+II and Joint Decoder II; Combined (oracle): the oracle combined system under the assumption that the SNR information is available; Combined I: the combined system based on the energy level estimation using the deep denoisers; Combine II: the combined system based on the confidence level estimation.

Systems	Conditions						AVG
	6dB	3dB	0dB	-3dB	-6dB	-9dB	
DNN	32.5	48.8	66.3	78.4	86.3	91.8	67.4
IBM [1]	15.4	17.8	22.7	20.8	22.1	30.9	21.6
DNN I+II	**4.5**	16.9	49.8	39.8	21.7	**19.6**	25.4
Joint Decoder II	16.5	17.1	19.9	18.8	22.5	25.3	20.0
Combined (oracle)	4.5	16.9	19.9	18.8	21.7	19.6	16.9
Combined I	16.0	16.6	**19.7**	18.8	23.0	24.1	19.7
Combined II	11.1	**15.9**	22.5	**21.3**	**20.7**	21.3	**18.8**

4.5.7 System Combination Using Confidence Scores

As shown in Section 4.5.4, the denoised features have severe residuals when the two mixed speech signals have similar energy levels, which will leads to very inaccurate estimations of the energy ratio. Therefore, we propose another approach to do the system combination based on the confidence score derived from the decoded lattice. The main idea is that if one system generates the results with the high sequential posteriors $p(W|x_{1:T})$, the system should work better with relatively high confidence. So we can use it to determine whether we use DNN I+II or Joint Decoder system. The sequential posteriors are derived from the decoded lattices using the equation,

$$p(W|x_{1:T}) = \frac{p(W, x_{1:T})p(W)}{\sum_{W'} p(W', x_{1:T})p(W')}. \tag{80}$$

We first use the posteriors generated by either high or low energy signal models alone. The assumption is that if one of two speakers in the speech signal dominates, either high or low energy signal model would generate the fairly high $p(W|x_{1:T})$. But we find the system combination based on this strategy does not work. Alternatively, we try to derive the sequential posteriors from the lattices generated by the joint decoder working with the instantaneous high and low energy models. Note that in this case each path in the lattice will contain the state sequences for both speakers. For the efficient evaluation, we use the

following equation as the sequential posteriors for two speakers together,

$$C = \frac{p(W^1, x_{1:T})p(W^1)p(W^2, x_{1:T})p(W^2)}{\sum_{W'^1,W'^2} p(W'^1, x_{1:T})p(W'^1)p(W'^2, x_{1:T})p(W'^2)}. \quad (81)$$

With this confidence scores, we first tune and obtain a optimal threshold on the development set, and use it for the system combination. The results are listed in Table 18. In this case, we obtain the lowest 18.8% overall WER.

4.6 Summary

In this chapter, we investigate DNN-based systems for single-channel mixed speech recognition by using multi-style training strategy. We also introduce a WFST-based joint decoder to work with the trained DNNs. Experiments on the 2006 speech separation and recognition challenge data demonstrate that the proposed DNN based system has remarkable noise robustness to the interference of competing speaker. The best setup of our proposed systems achieves 18.8% overall WER which improves upon the results obtained by the IBM superhuman system by 2.8% absolute, with making fewer assumptions and lower computational complexity.

Our next work will focus on how we can better track one speaker in the multi-talker scenarios. In fact we have tried DNN-based unsupervised mask learning for one speaker based on the entropy criteria. Although this approach has not worked yet, recent approach proposed in [107] indicates that a good initialization appears to very critical to learn a well-perform mask. Therefore, to learn a good mask for tracking certain speaker will be one important perspective of our future work.

CHAPTER 5

LATENT SEMANTIC RATIONAL KERNELS FOR TOPIC SPOTTING ON SPEECH

5.1 Introduction

Topic spotting aims at automatically determining the topic of a given speech utterance, and can be considered as a classification problem if the topic to be determined is among a fixed set. Most of the previous works deal with this problem by first decoding the given speech utterance into a word transcription and then treating it as a document categorization problem, thus many existing text analysis techniques can be applied. In [108], a set of keywords are first selected according to their relative contributions to the discrimination of topics and topic spotting is then employed by scoring the decoded transcription based on the selected keywords. A similar idea has been applied in the famous AT&T HMIHY call-routing task [35], in which a set of *salient words or phrases* [109] are chosen based on their relative mutual information with certain call-types, and then the call is classified according to the detected salient word or phrase. More recently in [110], topic spotting with a more sophisticated document classification algorithm, namely the BOOSTEXTER, was explored, together with a special learned grammar for the automatic speech recognition (ASR) decoding.

A common drawback of these methods is that the topic spotting strategy is still based on the 1-best ASR decoded word transcription, which may not be sufficiently reliable to deliver a good topic classification performance in some challenging tasks, *e.g.*, spontaneous conversational speech, the word recognition accuracy of which is in general rather low. To overcome this, researchers began to extract linguistic features from the phoneme or word lattices [111] [112] generated by the ASR decoding procedure for more robust topic spotting. Cortes et al. [113] proposed the rational kernels, which are a series of kernels built upon the weighted finite-state transducers (WFSTs). Classification can be conducted via

support vector machine (SVM) using rational kernels based on WFSTs (lattices) which compactly represent all the most likely transcriptions from the ASR outputs. This framework is fairly appealing in that many well-defined and efficient operations of WFSTs can be taken advantage of to implement the feature space mapping and inner product for training the SVM. Among all the rational kernels that have the positive definite and symmetric (PDS) property [113], the n-gram rational kernel is prevalent in topic spotting applications. The approach typically first maps the WFSTs to a high dimensional n-gram feature space and then employs an inner product for topic identification. However, the n-gram rational kernel assumes an exact match of the n-grams and treats the contribution of each n-gram (words or phrases) to the topic discrimination uniformly, resulting in a substantial degradation in the topic spotting performance especially for spontaneous speech in which filler or functional words frequently appear and interfere with the actual discriminability.

To handle this issue of n-gram rational kernels, in our earlier work [114], we proposed the latent semantic rational kernels (LSRK) for topic spotting on spontaneous speech and showed that topic spotting with the LSRK framework gave very promising results. In this chapter, we extend our earlier work by incorporating more thorough derivations of the LSRK framework and presenting more details about its generalization using additional text analysis techniques including probabilistic topic models. In the LSRK framework, rather than mapping the WFSTs onto an n-gram feature space, we map the WFSTs onto a reduced dimensional latent semantic space as in latent semantic analysis (LSA) [115] or probabilistic topic models. Compared to the n-gram rational kernels, LSRK needs another layer of WFST composition with mapping transducers to map the original n-gram features to the low-dimensional representations. We will discuss how to generalize LSRK in two separate forms of transducer such that all available external knowledge and techniques can be flexibly incorporated into the proposed LSRK framework to enhance the topic spotting performance. To be specific, we will present several examples to demonstrate the use of the LSRK framework incorporated with several typical text analysis techniques such as tf-idf,

WordNet, LSA or the combinations of them. We further show that the family of n-gram rational kernels is a special case of LSRK when the term similarity matrix is an identity matrix. Then we focus on how to generalize the LSRK using probabilistic topic models, *e.g.*, PLSA [116] or LDA [71]. To demonstrate the effectiveness of the LSRK framework, we conduct topic spotting experiments using SVM with LSRKs on two datasets, the Switchboard and AT&T's HMIHY0300. Experimental results show that using SVMs with LSRKs can achieve significant gain in topic spotting performance over the baseline n-gram rational kernels in both datasets.

The remainder of this chapter is organized as follows: Section 5.2 gives an overview of WFSTs and n-gram rational kernels, which serves as the preliminaries and background of this chapter. We give the formulations of LSRK in 5.3. Then we focus on how to generalize the LSRK framework in 5.4, especially the generalization of LSRK using probabilistic topic models. Experimental results are reported in Section 5.5 and finally we conclude our work in Section 5.6.

5.2 N-gram Rational Kernels

In this section, we present necessary WFSTs algebraic definitions and notations to introduce rational kernels and describe the n-gram rational kernel.

5.2.1 WFSTs and Rational Kernels

A system $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is a semiring if : $(\mathbb{K}, \oplus, \overline{0})$ is a commutative monoid with identity element $\overline{0}$; $(\mathbb{K}, \otimes, \overline{1})$ is a monoid with identity element $\overline{1}$; $\otimes$ distributes over $\oplus$; and $\overline{0}$ is an annihilator for $\otimes$ (for all $a \in \mathbb{K}$, $a \otimes \overline{0} = \overline{0} \otimes a = 0$). We list some commonly used semirings in Table 19. Two semirings that are often used in speech and language processing applications are the log semirings (similar to the probability semiring but with weight manipulation conducted in the negative log domain) and the tropical semirings (derived from the log semiring used for approximating Viterbi decoding). A WFST T [23] over a semiring $\mathbb{K}$ is an 8-tuple $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$, where Σ is the finite input alphabet of

Table 19: Commonly used Semirings. $\oplus_{\log}$ is defined by $x \oplus_{\log} y = -\log(e^{-x} + e^{-y})$.

SEMIRING	SET	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$
Boolean	$\{0, 1\}$	$\vee$	$\wedge$	0	1
Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	min	$+$	$+\infty$	0

the transducer, Δ is the finite output alphabet, Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times Q$ is a finite set of transitions, $\lambda : I \to \mathbb{K}$ is the initial weight function, and $\rho : F \to \mathbb{K}$ is the final weight function mapping F to $\mathbb{K}$. A weighted finite-state acceptor (WFSA) can be formally defined in a similar way but with the same input and output labels. Given a transition $e \in E$, we denote by $p[e]$ its origin or previous state and $n[e]$ its destination or next state, and $w[e]$ its weight. In the context of speech recognition, the input and output labels usually correspond to the HMM transition identifiers, mono-phones or tri-phones and words. The weight corresponds to either the acoustic or language score for a certain speech frame. A path $\pi = e_1 \cdots e_k$ consists of consecutive transitions, $n[e_{i-1}] = p[e_i]$, $i = 2, ..., k$, and a successful path in a WFST/WFSA is a path from an initial state to a final state with the weight as the $\otimes$-product of the weights of its constituent transitions, $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$. Let $P(q, q')$ be the set of paths from state q to q' and $P(q, x, y, q')$ the set of paths from q to q' with input label $x \in \Sigma$ and output label $y \in \Delta$, then the output weight associated through T to any pair of input-output string (x, y) is given by,

$$[\![T]\!](x, y) = \bigoplus_{\pi \in P(I,x,y,F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho[n[\pi]], \tag{82}$$

which is well defined in $\mathbb{K}$ and $[\![T]\!](x, y) = \overline{0}$ when $P(I, x, y, F) = \emptyset$. Given a weighted automaton or transducer M, the *shortest-distance* from state q to the set of final states F is defined as the $\oplus$-sum of all the paths from q to F,

$$d[q] = \bigoplus_{\pi \in P(q,F)} w[\pi] \otimes \rho[n[\pi]]. \tag{83}$$

Suppose now we want to decode a speech utterance and generate a lattice for it. Eq. (82) and (83) will be the main operations which require beam pruning during token passing on the WFST decoding graph; *i.e.*, find or retain the most likely word sequences and their corresponding state-level alignment.

For any transducer T, we denote by T^{-1} its *inverse*, which is the transducer formed by swapping the input and output labels of each transition and the input and output alphabets of T. For *composition*, let $T_1 = (\Sigma, \Delta, Q_1, I_1, F_1, E_1, \lambda_1, \rho_1)$ and $T_2 = (\Delta, \Omega, Q_2, I_2, F_2, E_2, \lambda_2, \rho_2)$ be two WFSTs defined over a commutative semiring $\mathbb{K}$ such that Δ, the output alphabet of T_1, coincides with the input alphabet of T_2. Then, the result of the composition of T_1 and T_2 is a weighted transducer $T_1 \circ T_2$ and for all input-output strings pair (x, y),

$$[\![T_1 \circ T_2]\!](x, y) = \bigoplus_{z \in \Delta} [\![T_1]\!](x, z) \otimes [\![T_2]\!](z, y). \tag{84}$$

Note that a transducer can be viewed as a matrix over the set $\Sigma \times \Delta$ and composition as the corresponding matrix-multiplication.

Let A be a weighted automaton defined over the semiring $\mathbb{K}$ and the alphabet Σ, B a weighted automaton defined over the semiring $\mathbb{K}$ and the alphabet Δ, a weighted transducer $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ over the semiring $\mathbb{K}$ and a function $\psi : \mathbb{K} \to \mathbb{R}$. Then the *rational kernels* $K(A, B)$ over A and B is given by,

$$K(A, B) = \psi\left(\bigoplus_{(x,y) \in \Sigma \times \Delta} [\![A]\!](x) \otimes [\![T]\!](x, y) \otimes [\![B]\!](y)\right), \tag{85}$$

for convenience, we use $w[M]$ as the shorthand for the *shortest distance* from the start state I to the set of final states F of the transducer M, (85) thus can be written as,

$$\begin{aligned} K(A, B) &= \psi\left(\bigoplus_{(x,y) \in \Sigma \times \Delta} [\![A \circ T \circ B]\!](x, y)\right) \\ &= \psi(w[A \circ T \circ B]) \end{aligned} \tag{86}$$

5.2.2 N-gram Rational Kernels

An N-gram kernel is a rational kernel that has PDS property [113] and has been widely and successfully used in speech or text classification applications [117]. Suppose A is a WFST

(word lattice) output from an ASR system, which evaluates a probability distribution P_A over all strings that can be represented by A, $s \in \Sigma$. Modulo a normalization constant, the weight assigned by A to a string x is $[\![A]\!](x) = -\log P_A(x)$ (for the log semiring). Denote by $|s|_x$ the number of occurrences of a sequence x in the string s. The expected count or number of occurrences of an *n-gram* sequence x in s for the probability distribution P_A is,

$$c(A, x) = \sum_s P_A(s)|s|_x. \tag{87}$$

The n-gram rational kernel k_n for two WFSTs A_1 and A_2 is defined as,

$$k_n(A_1, A_2) = \sum_{|x|=n} c(A_1, x)c(A_2, x), \tag{88}$$

which is typically the sum of products of the expected counts that A_1 and A_2 assign to their common *n-gram* sequences. In the WFST framework, n-gram rational kernels can be calculated efficiently as,

$$k_n(A_1, A_2) = w[(A_1 \circ T) \circ (T^{-1} \circ A_2)] = w[A_1 \circ (T \circ T^{-1}) \circ A_2], \tag{89}$$

where T is the transducer that can be used to extract all n-grams and compute $c(A_1, x)$,

$$T = (\Sigma \times \{\epsilon\})^* (\sum_{x \in \Sigma} \{x\} \times \{x\})^n (\Sigma \times \{\epsilon\})^*. \tag{90}$$

Fig.18 shows the T transducer in the case of bi-gram sequences ($n = 2$) and for the vocabulary $\Sigma = \{a, b\}$.

5.3 Latent Semantic Rational Kernels

In this section, based on the n-gram rational kernels, we propose the latent semantic rational kernels (LSRKs). We first give a formulation of the LSRK and briefly show how LSRK can be generalized to incorporate any form of external knowledge to enhance the topic spotting performance.

Recall that kernel methods first map the input to a high dimensional feature ϕ space,

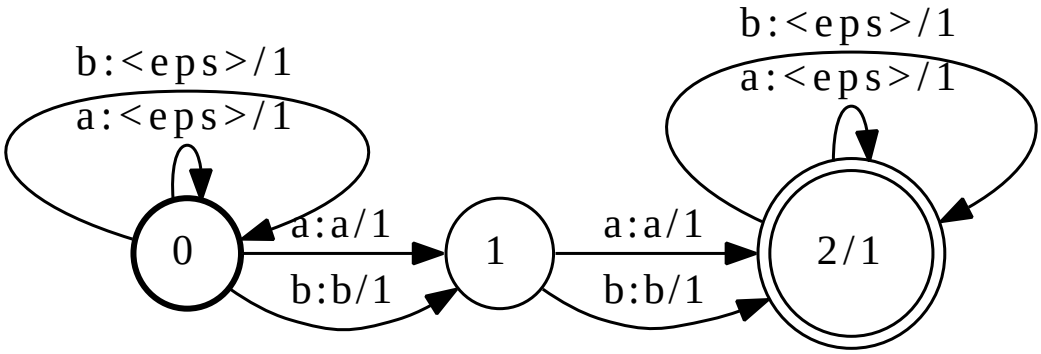

Figure 18: T transducer computing expected counts of bi-gram sequences of a word lattice with $\Sigma = \{a, b\}$, note that $\langle \text{eps} \rangle$ represents ϵ denoting the empty label

and take the inner product of them. Here we rewrite (88) as,

$$k_n(A_1, A_2) = \sum_{|x|=n} c(A_1, x)c(A_2, x) = \langle \phi(A_1), \phi(A_2) \rangle$$
$$= \phi(A_1)^T \phi(A_2), \tag{91}$$

where $\phi(A)$ is the mapped feature vector. We can see a WFST is first map to an n-gram space, and the value corresponding to each dimension is the expected count for this n-gram. It can be seen that there are two main limitations with n-gram rational kernels used for topic spotting: The N-gram kernel assumes that WFSTs from the same topic share many exact-matched n-grams, but in reality many n-grams are often correlated, sometimes synonymously. Furthermore, the produced WFSTs assume uniform contributions from the n-grams, but in reality we often observe many words that are not useful for topic discrimination, *e.g.*, filler or functional words. At the same time, some significant terms such as salient phrases in HMIHY that represent cogently certain topics may have the risk of being neglected in the evaluation process.

If we treat WFST as *a distribution over multiple documents*, the ideas of both LSA and latent semantic kernels (LSK) [118] can be applied here naturally. In LSA, a document is first represented by a vertical vector d indexed by the terms in the vocabulary, and the corpus is then represented by a term-document matrix D, whose columns are indexed by the documents and whose rows are indexed by the terms, $D = [d_1, ..., d_m]$. If we define the

kernels over two documents as,

$$K(d_1, d_2) = \langle d_1, d_2 \rangle = d_1^T d_2, \tag{92}$$

it becomes similar to the n-grams rational kernels over WFST, which measures the similarity by counting exact matches in terms/n-grams. But as in LSA or LSK, d will be first mapped into a latent semantic space to explore the semantic relationship between terms. This space with a much reduced dimensionality is projected via singular value decomposition (SVD) on the D matrix. Denote by $\mathcal{T}$ the linear transform we use to map d to the latent semantic space, the latent semantic kernel is defined as,

$$K(d_1, d_2) = \langle \mathcal{T} d_1, \mathcal{T} d_2 \rangle = d_1^T \mathcal{T}^T \mathcal{T} d_2. \tag{93}$$

Similarly, for the n-gram rational kernels, we can modify (91) to,

$$k_n(A_1, A_2) = \langle \mathcal{T}\phi(A_1), \mathcal{T}\phi(A_2) \rangle = \phi(A_1)^T \mathcal{T}^T \mathcal{T} \phi(A_2). \tag{94}$$

Since we do not always have to express the feature vector explicitly (kernel trick), we define the *Latent Semantic Rational Kernels* (LSRK) as,

$$k_n(A_1, A_2) = \langle \mathcal{T}\phi(A_1), \mathcal{T}\phi(A_2) \rangle = \phi(A_1)^T \mathcal{S} \phi(A_2). \tag{95}$$

Compared to the basic n-gram rational kernels, we only need to multiply the feature vector by one matrix $\mathcal{S}$ before computing the inner product, which implies another WFST composition operation. In the WFST framework, suppose S is the WFST representing the matrix $\mathcal{S}$. The LSRK can be calculated as,

$$\begin{aligned} k_n(A_1, A_2) &= w[(A_1 \circ T) \circ M \circ M^{-1} \circ (T^{-1} \circ A_2)] \\ &= w[(A_1 \circ T) \circ S \circ (T^{-1} \circ A_2)], \end{aligned} \tag{96}$$

where M is the transducer encoding the transform which maps the original n-gram features to the low representations; S WFST can be defined as,

$$S = (\{\epsilon\} \times \{\epsilon\})^* (\sum_{x \in \Sigma} \{x\} \times \{x\})^n (\{\epsilon\} \times \{\epsilon\})^*, \tag{97}$$

One example of the S transducer in the bi-gram case is shown in Fig.19, in which each arc corresponds to the elements in the $\mathcal{S}$ matrix; *e.g.*, $\mathcal{S}(i, j)$ corresponds to the arc with input label i, output label j and weight $\mathcal{S}(i, j)$. Then, the S for n-gram LSRK is constructed by concatenating n stages like this. S may appear to contain a large number of arcs, $n\times|\Sigma|\times|\Sigma|$, but in reality $\mathcal{S}$ can be very sparse over the non-diagonal elements and is thus still tractable after we use some heuristics to prune it.

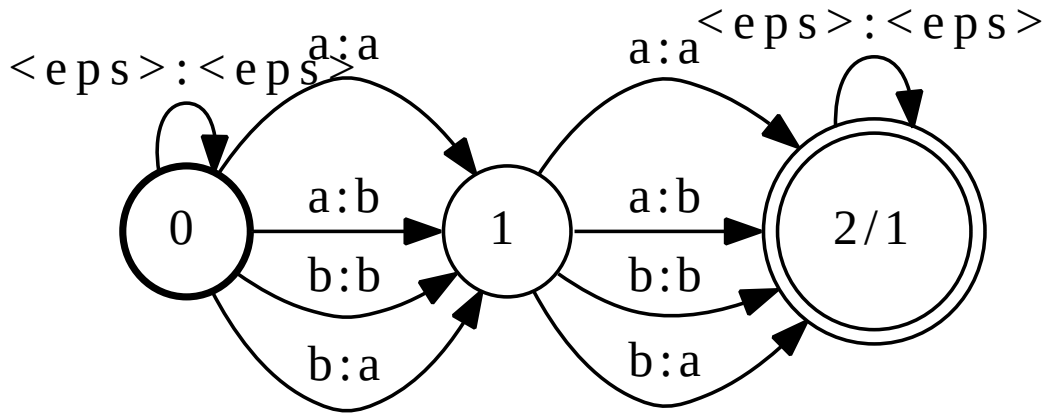

Figure 19: S transducer (without weight on arcs) computing LSRK of a word lattice with $\Sigma = \{a, b\}$ (bi-gram case)

If we take an insightful look at the $\mathcal{S}$ matrix as in (95), it actually can be viewed as the *term-term similarity matrix* which specifies the *semantic similarity* between terms, *e.g.*, the value of element $\mathcal{S}(i, j)$ measures the semantic similarity between terms i and j. In the n-gram rational kernels case, it assumes the semantic similarity of the same term is 1 and there exists no semantic similarity between different terms which corresponds to the special case of LSRK with $\mathcal{S}$ being set to an identity matrix I. This motivates us to generalize the LSRK with respect to the term-term similarity matrix $\mathcal{S}$. That is, $\mathcal{S}$ is not necessarily constructed from the LSA; instead, it can be designed in multiple ways such that any form of available external knowledge can be incorporated into it. This generalization gives us many possibilities to use LSRK. We briefly list several typical cases to use LSRK as illustrations.

- If $\mathcal{S} = I$, LSRK is equivalent to the n-gram rational kernels.

- If $\mathcal{S} = \text{diag}\left(\text{idf}^2(1), ..., \text{idf}^2(i), ..., \text{idf}^2(N)\right)$, where $\text{idf}(i)$ is the *inverse document frequency* of term i according to the training corpus. In this case, LSRK will count the expected tf-idfs (term frequency-inverse document frequency) [119] assigned to the common n-grams. Note that the expected term frequency is already evaluated with $A \circ T$ part in (96), we only need idfs for each term in the matrix $\mathcal{S}$.

- If $\mathcal{S} = U_K \Sigma_K^{-1} \Sigma_K^{-1} U_K^T$, where U_K and Σ_K are the corresponding matrices obtained from the K-rank approximation to the term-document matrix using SVD as in LSA, $D \approx U_K \Sigma_K V_K^T$. In this case, the $\mathcal{S}$ is constructed from the latent semantic space in a data-driven way.

- If $\mathcal{S}_{ij} = \text{WordNet :: Similarity}(i, j)$, the $\mathcal{S}$ matrix is then constructed from the WordNet ontology [120]. Various algorithms [121] using WordNet can be used to determine the similarity; this approach models the similarity based on the distance between the conceptual categories of words and the hierarchical structure in the WordNet.

In real applications, several techniques can be combined to obtain an effective $\mathcal{S}$ matrix. The training corpus we use to estimate the matrix $\mathcal{S}$ is not limited to the speech transcriptions which usually are limited and expensive. With LSRK, more available text corpus can be utilized to boost the topic spotting performance.

5.4 Generalization of Latent Semantic Rational Kernels

In the last section, we briefly mentioned several examples to generalize the LSRK. Recall that there are two forms regarding the formulations of the LSRK,

$$\begin{aligned} k_n(A_1, A_2) &= w[(A_1 \circ T) \circ M \circ M^{-1} \circ (T^{-1} \circ A_2)] \\ &= w[(A_1 \circ T) \circ S \circ (T^{-1} \circ A_2)]. \end{aligned} \tag{98}$$

The first one is based on the transducer M which encoding the transform which transforms certain WFST n-gram features into certain low dimensional representations; the second

is based on the transducer S which encodes the term-term similarity information. In this section, we will focus on the generalization of LSRK in terms of these two interpretations of LSRK based on the different definitions of the transducer M or S.

5.4.1 LSRK Generalization using Vector Space Models

Vector space models (VSM) [122] represent documents as vectors of indexed words and evaluate the similarity between documents using a cosine similarity measure in an algebraic framework. Thus LSRK generalization with VSM will be fairly straightforward. In this subsection we will discuss LSRK with tf-idf weighting, LSA, WordNet and LSA with semantic expansion (or semantic smoothing) using WordNet.

5.4.1.1 LSRK with tf-idf weighting

Tf-idf is a widely used term weighting strategy in a text retrieval task. The tf-idf value increases proportionally to the number of occurrences in the documents, while being offset by the frequency of the word in the whole corpus,

$$\text{tf} - \text{idf}(w, d, D) = \text{tf}(w, d) \times \text{idf}(w, D), \tag{99}$$

where $\text{idf}(t, D)$ is the inverse document frequency,

$$\text{idf}(w, D) = \log \frac{|D|}{|\{d \in D : w \in d\}|}. \tag{100}$$

We have briefly mentioned the LSRK with tf-idf weighting in Section 5.3. For the LSRK, since the $\text{tf}(w, A)$ part is already evaluated with the $A \circ T$ operation, we only need to construct a transducer S_{idf}, which contains two states and V arcs between them. The input and output labels on each arc are both word index j for w_j, and the weight is the square of the inverse document frequencies $\text{idf}^2(w_j)$. Then the LSRK with tf-idf weighting is given by,

$$K(A_1, A_2) = w\left[A_1 \circ T \circ S_{\text{idf}} \circ T^{-1} \circ A_2\right]. \tag{101}$$

5.4.1.2 LSRK with LSA

In the LSA, documents are represented as vectors in a low dimensional latent semantic space. Suppose a document is first represented by a vertical vector d indexed by the terms

in the vocabulary. The corpus is represented by a term-document co-occurrence matrix D, whose columns are indexed by the documents and whose rows are indexed by the terms, $D = [d_1, ..., d_m]$. Then SVD is employed on the term-document co-occurrence matrix D,

$$D = U\Sigma V^T, \tag{102}$$

and then select the K most significant spanning values from Σ to form Σ_K together with their corresponding singular vectors from U to form U_K. The transforms used for mapping the original vector into the latent semantic space is thus given by,

$$\mathcal{T} = U_K \Sigma_K^{-1}. \tag{103}$$

We first define a tranducer M_{LSA} which encodes the transform $\mathcal{T}$: M_{LSA} contains two states and $V \times K$ arcs between them. The input labels of each arc represent words indices and output labels represent the dimensional indices in the mapped latent semantic space. The weight on the arc with input and output labels pair (i, k) will be $\mathcal{T}(i, k)$. Therefore the LSRK with LSA can be written as,

$$\begin{aligned} K(A_1, A_2) &= w\left[A_1 \circ T \circ M_{\text{LSA}} \circ M_{\text{LSA}}^{-1} \circ T^{-1} \circ A_2\right] \\ &= w\left[A_1 \circ T \circ S_{\text{LSA}} \circ T^{-1} \circ A_2\right]. \end{aligned} \tag{104}$$

From the equation above, another form of the LSRK with LSA is to regard $M_{\text{LSA}} \circ M_{\text{LSA}}^{-1}$ as one transducer S_{LSA}. An example of S_{LSA} is given in Fig.19. The form using M_{LSA} usually leads to a more efficient implementation, while the other form using S_{LSA} is more appealing in terms of its term-term similarity suggesting that the LSRK with LSA can be easily combined with other schemes, *e.g.*, tf-idf weighting or WordNet.

5.4.1.3 LSRK with WordNet

WordNet is a large lexical database which groups words into synonyms sets (synsets) and then records various semantic relations between these synsets. One important use of WordNet is to determine the semantic similarity between words. In this regard, various algorithms have been proposed based on the distance between the conceptual categories of

synsets and the hierarchical structure in the WordNet ontology. For example, Resnik [123], Lin [124] and Jiang et al. [125] measure similarity between two words based on the information content of their least common subsumer in the WordNet where the information content is a measure of the specificity of a concept. Some other measures like Leacock et al. [126] and Wu et al. [127] usually first find the shortest path between two concepts and evaluate the similarity between words using the normalized path length.

Suppose now we have the matrix $\mathcal{S}_{\text{WN}}$. Each entry (i, j) gives the semantic similarity between the terms i and j according to WordNet using certain measures. Although the similarity measure we described above is all between two single words, the matrix can incorporate higher order n-gram features by using WordNet-based similarity measures for the phrases as in [128]. After encoding the matrix $\mathcal{S}_{\text{WN}}$ into the transducer S_{WN}, the LSRK with WordNet can thus be given by,

$$K(A_1, A_2) = w\left[A_1 \circ T \circ S_{\text{WN}} \circ T^{-1} \circ A_2\right]. \tag{105}$$

5.4.1.4 Semantic Expansion using WordNet for LSRK with LSA

Another way of using WordNet for LSRK is to do the semantic expansion (a.k.a semantic smoothing as in [118]) with it for LSA. Speech transcriptions are usually used as the training corpus for topic spotting; however, in some cases documents (transcriptions for each utterance) appear to be too sparse (with insufficient meaningful words) to obtain effective similarity matrix. This issue can be mitigated by employing semantic expansion using knowledge based WordNet ontology. Denote by $\mathcal{S}_{\text{WN}}$ the similarity matrix obtained from WordNet used for semantic expansion. Then the semantic expanded version of the document d is given by,

$$d_{\text{ex}} = \mathcal{S}_{\text{WN}} \times d. \tag{106}$$

Rather than performing SVD on D, we do LSA on the semantic expanded term-document matrix,

$$D_{\text{ex}} = \mathcal{S}_{\text{WN}} \times D. \tag{107}$$

The left part of LSRK formulation is just the same as LSRK with LSA.

5.4.2 LSRK Generalization using Probabilistic Topic Models

Compared to the vector space models, probabilistic topic models, which are usually based on the likelihood principle and define a proper generative model of the data, have more solid statistical foundation. In contrast, one important benefit of the probabilistic topic models is to discover the latent topic structure among a large archive of documents. In probabilistic topic models, documents are usually modeled as the mixtures over latent topics where each topic is characterized by a distribution over words. Probabilistic latent semantic analysis (PLSA) [116] and latent Dirichlet allocation (LDA) [71] are most typical algorithms for the probabilistic topic models. In this subsection, we first review some key aspects of these two topic modeling techniques and then focus on how we can generalize LSRK utilizing learned latent topics from them.

5.4.2.1 Probabilistic Latent Semantic Analysis and its Applications on Document Categorization

Evolved from LSA, PLSA introduces a bottleneck latent variable z. The graphical model representations of PLSA are shown in Fig.20 where the observations are in the form of co-occurrence of words and documents (w_i, d_j), the probability of each co-occurrence is modeled as a mixture of conditionally independent multinomial distributions. For each document d, a latent topic z_k is chosen conditionally according to $P(z_k|d)$ and a word is then generated from that topic according to $P(w|z_k)$,

$$P(w, d) = P(d) \sum_{k=1}^{K} P(z_k|d)P(w|z_k). \tag{108}$$

where $P(w|z_k)$ is the multinomial factor distribution corresponding to a certain latent class (topic) k on words, and $P(z_k|d)$ is the document-specific mixing weight for topic z_k. To learn a PLSA model, both $P(w|z_k)$ and $P(z_k|d)$ are estimated iteratively via EM algorithm, for the E-step,

$$P(z_k|d_j, w_i) = \frac{P(z_k)P(d_j|z_k)P(w_i|z_k)}{\sum_{k'=1}^{K} P(z_{k'})P(d_j|z_{k'})P(w_i|z_{k'})}, \tag{109}$$

for the M-step,

$$\left.\begin{array}{c} P(d_j|z_k) \\ P(w_i|z_k) \\ P(z_k) \end{array}\right\} \propto \sum_{j'=1}^{D}\sum_{i'=1}^{V} n(d_{j'}, w_{i'}) P(z_k|d_j, w_i) \times \left\{\begin{array}{c} \delta_{jj'} \\ \delta_{ii'} \\ 1 \end{array}\right. , \tag{110}$$

where $n(d_j, w_i)$ is the count how often a word w_i occurred in a document d_j and δ denotes the Kronecker delta. Note that in the EM iterations above, the model is equivalently parameterized in the form as shown in Fig.20 (b).

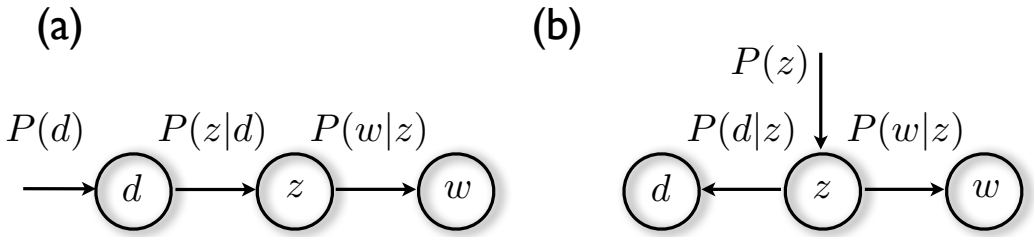

Figure 20: Two graphical model representations of PLSA: (a) asymmetric parameterization (b) symmetric parameterization

PLSA has been applied in document retrieval tasks and shows its advantage over LSA in [129]. In that work, Hofmann provided two ways of using PLSA for computing similarities between documents. One is to represent a document using an interpolated document specific language model (LM) as,

$$P(w_i|d_j) = \mu\hat{P}(w_i|d_j) + (1-\mu)P_{\text{PLSA}}(w_i|d_j), \tag{111}$$

where $\hat{P}(w_i|d_j)$ is the ML estimated document LM and $P_{\text{PLSA}}(w_i|d_j)$ is PLSA estimated LM. The other is using the low-dimensional representation $P(z_k|d_j)$ to evaluate similarities. Both ways need a folding-in process (computing a representation for a document or query that is not in the original training set). This is accomplished via EM iterations while keeping the learned topic distribution $P(w_i|z_k)$ fixed as:

$$\text{E step}: \quad P(z_k|d_j, w_i) = \frac{P(w_i|z_k)P(z_k|d_j)}{\sum_{k=1}^{K} P(w_i|z_k)P(z_k|d_j)}, \tag{112}$$

$$\text{M step}: \quad P(z_k|d_j) = \frac{\sum_{i=1}^{V} n(d_j, w_i)P(z_k|d_j, w_i)}{n(d_j)}. \tag{113}$$

These two equations are very important as they are the key formulas we will use for generalizing LSRK using probabilistic topic models and we will see they can be evaluated efficiently via the WFST operations.

A more rigorous usage of PLSA for document categorization tasks is proposed in [130]. In that work, Fisher kernels [131] are derived based on the PLSA models as two parts, one being

$$K_1(d_j, d_l) = \sum_{k=1}^{K} P(z_k|d_j)P(z_k|d_l)/P(z_k), \tag{114}$$

the other is,

$$K_2(d_j, d_l) = \sum_{i} \lambda_i \hat{P}(w_i|d_j)\hat{P}(w_i|d_l), \tag{115}$$

where

$$\lambda_i = \sum_{k=1}^{K} \frac{P(z_k|d_j, w_i)P(z_k|d_l, w_i)}{P(w_i|z_k)}. \tag{116}$$

The Fisher kernel based on PLSA is then the additive combination of two kernels K_1 and K_2.

5.4.2.2 Latent Dirichlet Allocation

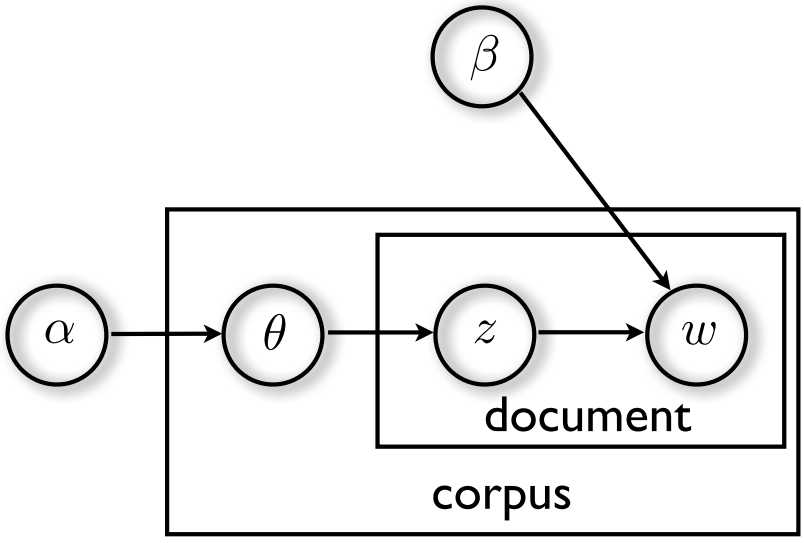

Figure 21: The graphical model representation of LDA: α is the parameter of the Dirichlet prior on the documents' topic distributions; β is the parameters matrix representing per-topic word distributions; z is latent variable represents the topic.

A potential issue with PLSA is that the number of parameters in the model grows with the number of documents in the corpus, which could lead to overfitting. LDA is a generalization of PLSA by introducing a global Dirichlet prior on the topic distribution for each document. A graphical model representation of LDA is shown in the Fig21, where z is the latent variable denotes potential topic. Rather than drawn from a document-specific distribution over topics in PLSA, z is drawn from a multinomial distribution parameterized by θ which is again drawn from a global Dirichlet prior parameterized by α; β is a parameter matrix where $\beta(i,k)$ denotes $P(w_i|z_k)$. The likelihood of a document given the parameters α and β is usually given in a fully Bayesian fashion by integrating out the θ and z as,

$$p(d_j|\alpha,\beta) = \int P(\theta|\alpha)\left(\prod_{i=1}^{N}\sum_{z_k} P(z_k|\theta)p(w_i|z_k,\beta)\right)d\theta. \tag{117}$$

Since the EM formulas for LDA can not be expressed analytically, the parameters estimation is usually done via variational EM iterations or some sampling based approaches; for more details please consult [71] and [132].

Document representations based on LDA for both document classification and retrieval tasks have been presented in the original LDA work and [133], respectively. In both works, computing the representations for the documents that are not in the training set needs new rounds of approximate inference, i.e, variational EM or Gibbs sampling, which usually needs quite a few iterations to reach convergence. This is fairly undesirable for a real-time topic spotting system. We will use the same folding-in process as in (112) and (113) regardless the topic models are learned from PLSA or LDA for two reasons: First, a small number of iterations are typically enough for folding-in; second, the relatively straightforward form of the EM formulas in folding-in makes efficient evaluation of the document representations via WFST operations practical.

5.4.2.3 LSRK with Probabilistic Topic Models

In [134], the relation between PLSA and non-negative matrix factorization (NMF) was shown which indicates that ML solution of PLSA is a solution of NMF with KL divergence

on the normalized term-document co-occurrence matrix. In this light, probabilistic topic models with K topics can be interpreted as mapping an original document with a fairly high dimensionality V to a point on a $K-1$ dimensional simplex. This geometric interpretation motivates us to generalize LSRK using probabilistic topic models.

The very first thing is how to map the input WFST A onto its low dimensional representation $P(z_k|A)$ according to the learned topic models. As aforementioned, the key formulas for this mapping procedure is the folding-in process defined in (112) and (113). Suppose we already learned a topic model using PLSA or LDA from a training corpus, *i.e.*, $P(w_i|z_k)$. (Note that we will keep this fixed during the folding-in process.) Define a WFST M as follows: there are two states within M, the start state and the final state with 1 as the final weight; the number of arcs between the these two states will be $V \times K$ and for each arc the input label is the word index i for w_i and the output label is latent class index k for z_k, and the weight is the corresponding topic model parameter $P(w_i|z_k)$. An example of WFST M is shown in Fig.22, where the vocabulary is shown as a, b, c ($V = 3$) and there are two latent topics ($K = 2$). Note that the weights of those arcs with identical output labels are summed up to 1. As can be seen, the M transducer encodes all the information we need from the learned topic models.

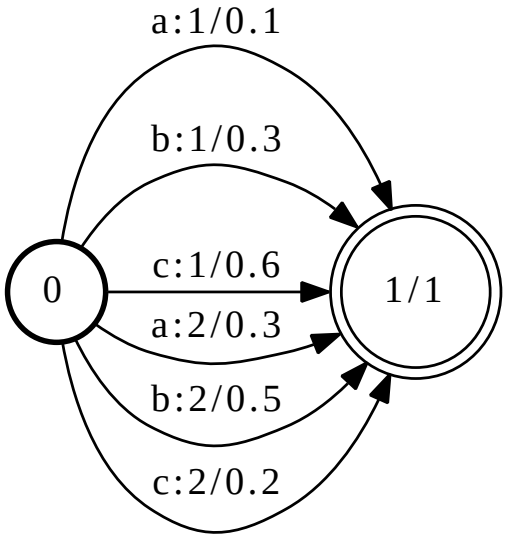

Figure 22: M transducer with $\Sigma = \{a, b, c\}$ and two latent topics $K = 2$.

Denote by $W^{(0)}$ the acceptor encoding the initial values of the low dimensional representation $P(z_k|A)$ for the input WFST A, which contains two states, and K arcs between them, with input and output labels are topic index k and the weights are the corresponding topic mixture weights $P(z_k|A)$. Having the transducers M and $W^{(0)}$ in hand, we can evaluate the E-step for the folding-in as in (112) with the composition operation between M and W_0, and the only step left is to divide the weight of each arc in the resultant transducer by the accumulated weight of those arcs with the same corresponding output label. Denoted by $\mathrm{DIV}(\cdot)$ this dividing operation on the weights, the E-step for the re-estimation of $P(z_k|A)$ at the tth iteration can be evaluated as,

$$M^{(t)} = \mathrm{DIV}(M \circ W^{(t-1)}). \tag{118}$$

For the M-step, notice that the $\frac{n(d_j,w_i)}{n(d_j)}$ part is the number of word w_i occurring in document d_j normalized by the length of d_j. Recall that in n-gram rational kernels, the expected count for w_i in A can be easily evaluated using $A \circ T$. The only difference is we need to normalize the words' *expected* counts by the *expected* length of A (remember that a transducer A can be regard as a *distribution* over multiple documents), which actually can be easily done by dividing the final weight of $A \circ T$ by the shortest distance from the start state to the final states,

$$\rho[F(A \circ T)] := \frac{\rho[F(A \circ T)]}{d[I(A \circ T)]} \tag{119}$$

For the sake of conciseness, we will not express the above operation explicitly in our formulations. Finally, the M-step at the tth iteration can be evaluated as,

$$W^{(t)} = \pi_o(A \circ T \circ M^{(t)}), \tag{120}$$

where $\pi_o(\cdot)$ is the projection operation which projects an WFST onto its output label domain copying each arc's input label to its output label. We summarize the whole procedure in Algorithm 3.

With $W^{(T)}$ available, LSRK corresponding to the first part of the Fisher kernels as in

Algorithm 3 EM Re-estimation of $P(z_k|A)$

Input: input acceptor A, parameters $P(w_i|z_k)$ from the learned probabilistic topic models.

1: Encoding $P(w_i|z_k)$ into WFST M as described.
2: Initialize $P(z_k|A)$ and encoding it into the acceptor $W^{(0)}$.
3: **for** $t = 1, ..., T$ **do**
4: E step : $M^{(t)} := \text{DIV}(M \circ W^{(t-1)})$
5: M step : $W^{(t)} := \pi_o(A \circ T \circ M^{(t)})$
6: **end for**

Output: $W^{(T)}$, which encodes the re-estimated $P(z_k|A)$.

(114) can be expressed as (for simplification, we omit the factor $P(z_i)$),

$$\begin{aligned} K_1(A_1, A_2) &= w\left[W_1^{(T)} \circ W_2^{(T)}\right] \\ &= w\left[A_1 \circ T \circ M_1^{(T)} \circ (M_2^{(T)})^{-1} \circ T^{-1} \circ A_2\right] \end{aligned} \tag{121}$$

Deriving the LSRK corresponding to the second part of the Fisher kernels in (115) is a bit tricky. The form of the second part in general is the common word-occurrences weighted by λ_i. Since the common word-occurrences can be evaluated via n-gram rational kernels, the difficult part is the evaluation of λ_i. Noticing that the denominator part of λ_i is the statistics we obtain in the E-step of folding-in, which are encoded into the transducer $M^{(t)}$ as in (118). The key idea is to modify the transducer $M^{(t)}$ such that λ_i can be efficiently evaluated under WFST. The modification to $M^{(t)}$ is as follows. For each arc with input and output label (i, k), we divide its weights by $\sqrt{P(w_i|z_k)}$ and then remap the output label using the following rule,

$$k := (i - 1) \times K + k. \tag{122}$$

Denoted by $\mathcal{M}^{(T)}$ this modified transducer from $M^{(T)}$, the LSRK corresponding to the second part of the Fisher kernels can thus be written in the same form as the first part with $M^{(T)}$ replaced by $\mathcal{M}^{(T)}$,

$$K_2(A_1, A_2) = w\left[A_1 \circ T \circ \mathcal{M}_1^{(T)} \circ (\mathcal{M}_2^{(T)})^{-1} \circ T^{-1} \circ A_2\right] \tag{123}$$

Finally the generalized Fisher LSRK based on the topic models is given by,

$$K_{\text{Fisher}}(A_1, A_2) = K_1(A_1, A_2) + K_2(A_1, A_2) \tag{124}$$

Note that although we only give the full derivation of the generalized Fisher LSRK, the use of LSRK in the context of probabilistic topic models is not limited to this. Considering if we want to use the kernels based on the document representation in the form of (111) as in the original PLSA work, *i.e.*,

$$K(A_1, A_2) = \sum_i P(w_i|A_1)P(w_i|A_2), \tag{125}$$

where

$$P(w_i|A) = \mu\hat{P}(w_i|A) + (1-\mu)P_{\text{PLSA}}(w_i|A), \tag{126}$$

for the first part $\hat{P}(w_i|A)$, this can be easily evaluated using $A \circ T$. For the second part which has the form $P_{\text{PLSA}}(w_i|A) = \sum_k P(z_k|A)P(w_i|z_k)$, after having available the transducers M and $W^{(T)}$ which encode $P(w_i|z_k)$ and $P(z_k|A)$ respectively, the transducer A_{PLSA} which encodes the second part can be also easily evaluated as,

$$A_{\text{PLSA}} = \pi_I(M \circ W^{(T)}). \tag{127}$$

5.5 Experiments

In this section, to validate the proposed LSRK framework, we conduct the topic spotting experiments on two tasks, SwitchBoard-1 Release 2 and AT&T HMIHY0300. We will show that on both tasks, LSRK can achieve significant spotting performance gains over n-gram rational kernel baselines.

5.5.1 Experiments on Switchboard

We first evaluated the proposed LSRK framework for topic spotting on a challenging conversational telephone speech task, Switchboard-1 Release 2, which is a collection of 2438 two-sided telephone conversations among 543 speakers (302 males, 241 females). Each pair of callers is introduced a topic for discussion and there are about 70 topics.

5.5.1.1 The ASR system and WFSTs (lattices) Generation

We first describe the ASR system and how we generate the WFSTs (lattices) for each utterances. The ASR system is built using Kaldi Speech Recognition Toolkit [60], the acoustic

models are cross-word triphone models represented by 3-state left-to-right HMMs (5-state HMMs for silence) trained using MLE on about half data of the whole Switchboard corpus. A tri-gram language model (LM) is trained on the whole transcription of the dataset for decoding. The input features are MFCCs coupled with their linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT) and feature-space maximum likelihood linear regression (fMLLR) for speaker adaptation during later iterations. The WER of the ASR system on the HUB5 English evaluation set is 33.4%. With this ASR system, we first trained a uni-gram LM using the whole transcriptions of the dataset and then use it to generate lattices for around 100K utterances (about half of the whole dataset). These 100K WFSTs are the data we would use for the following topic spotting experiment.

5.5.1.2 Subset Selection for Topic spotting

It is found that a substantial amount of ill-formed utterances for topic spotting exist among those 100K utterances, *e.g.*, "UH, YEAH". We first filter out the filler words, functional words and stop words from the transcriptions for each utterance and then select utterances whose filtered transcriptions have appropriate length. (We set the length threshold to 20. The threshold is determined based on the trade-off between the length of each utterance and the number of speech utterances left.) Finally, with 20 as threshold there are around 10K utterances left. From those selected utterances, we filter out those topics that have less than 200 utterances. Finally, 4405 utterances on 19 topics are selected for the topic spotting tasks, and for each topic we randomly choose 90% for training and 10% for testing, as shown in Table 20.

5.5.1.3 Topic Spotting Using N-gram Rational Kernels

We use n-gram rational kernels as the baseline system. The topic spotting is conducted on this subset of Switchboard using multiclass SVMs with n-gram rational kernels and LSRK with the various generalizations respectively. For the n-gram rational kernels, we conduct the experiments with different n. Note that when $n > 1$, the kernels are actually obtained

Table 20: Number of utterances (train/test/total) for each topic in the subset of Switchboard used for the topic spotting evaluation

TOPIC	TRAIN	TEST	TOTAL
RECIPES/FOOD/COOKING	242	28	270
CAPITAL PUNISHMENT	197	23	220
PUBLIC EDUCATION	196	22	218
BUYING A CAR	207	24	231
PETS	204	23	227
WOMEN'S ROLE	191	22	213
TV PROGRAM	197	22	219
DIRECTIONS	245	28	273
GARDENING	200	23	223
WEATHER CLIMATE	250	28	278
MOVIES	193	22	215
GUN CONTROL	212	24	236
DRUG TESTING	193	22	215
AUTO REPAIRS	197	22	219
HOBBIES AND CRAFTS	188	21	209
EXERCISE AND FITNESS	230	26	256
AIR POLLUTION	180	21	201
CAMPING	186	21	207
RECYCLING	247	28	275
TOTAL	3955	450	4405

by taking the sum of all k_m in (88) as,

$$K_n = \sum_{m=1}^{n} k_m \quad 1 \leq m \leq n. \tag{128}$$

As shown in Table 21, we get 27.33% and 28.22% classification accuracy (which are comparable to the numbers reported on the Switchboard in [110]) in the unigram and bigram cases (we omit the results for higher n because the further improvements are marginal).

5.5.1.4 Topic Spotting Using LSRK with LSA and tf-idf weighting

In this setup, the way we generated term-term similarity $\mathcal{S}$ is a combination of LSA and tf-idf. We use each conversation transcription with those test utterances excluded as one document (2438 in total) to form the term-document matrix D. Note that the entries of the matrix are tf-idf weights of indexed words as defined in (99) and (100). Here the terms are not necessarily limited to the single words (uni-grams) in the vocabulary; one can simply extend each column vector d by adding the dimensions corresponding to the phrases with certain lengths, *i.e.*, bigrams, trigrams, etc. As incorporating the full high-order n-gram features in LSA will lead to a prohibitively large matrix D, certain pruning schemes on those high-order n-gram features have to be conducted in this case. Thus we leave the exploration of the high-order features for future study. We will only use uni-gram features. By employing SVD on this matrix, we obtain the initial matrix $\mathcal{S} = U_K \Sigma_K^{-1} \Sigma_K^{-1} U_K^T$. After this, we set the diagonal elements of $\mathcal{S}$ to the tf-idf weights of corresponding words and shrink or augment the non-diagonal elements proportionally. Denoted by $\mathcal{S}'$ this matrix, since $\mathcal{S}'$ is fairly large (over 30K×30K), we pruned those non-diagonal elements by selecting most N significant elements. Note that $\mathcal{S}'$ is symmetrical, so we can just focus on the upper-right half of the matrix, and choose most $N/2$ elements. With the pruned $\mathcal{S}'$, we compile it into transducers S to employ the LSRK. For the LSRK, we report the results in terms of different rank K for the LSA and the number of left non-diagonal elements N after pruning. As shown in Table 21, in all cases we obtain significant topic spotting gain (almost doubled) over n-gram rational kernels baseline. The effect of the different LSA dimensions on the

Table 21: Classification accuracies of LSRK with LSA and tf-idf wighting on the subset of Switchboard, N is the number of non-diagonal elements left in $\mathcal{S}$ after pruning, K is the rank for the low dimensional term-document matrix approximation in LSA.

System/Method	N (pruning)	K (LSA)	Accuracy
Unigram RK	-	-	27.33%
Bigram RK	-	-	28.22%
LSRK using LSA and tf-idf weighting	40K×2	500	52.44%
	80K×2	500	52.89%
	120K×2	500	52.44%
	160K×2	500	54.00%
	200K×2	500	53.78%
	1000K×2	500	56.67%
	40K×2	750	52.67%
	80K×2	750	52.44%
	120K×2	750	52.89%
	160K×2	750	53.56%
	200K×2	750	53.33%
	1000K×2	750	**57.56%**

spotting performance appears to be negligible. However, the different degrees of pruning of $\mathcal{S}'$ matters, we achieve further improvement in the less pruned cases and the highest one is 57.56% accuracy when we allow the number of non-diagonal elements to be $2000K$.

5.5.1.5 The Effect of Semantic Expansion Using WordNet

Now we look at the effect of semantic expansion using WordNet. First a term-term similarity matrix $\mathcal{S}_{\text{WN}}$ is obtained using WordNet based on lin measure [124], *i.e.*, $\mathcal{S}_{\text{WN}}(i, j) = \text{lin}(w_i, w_j)$. Note that when generating $\mathcal{S}$, for those words with the same lemma, they will have maximum similarity 1, i.e, we regard the word "recognize" and "recognizing" as the same word. After we obtain the term-document co-occurrence matrix D, this co-occurrence matrix is smoothed as $D_{\text{ex}} = \mathcal{S}_{\text{WN}} \times D$. Then we employ SVD on both the original and the semantic expanded co-occurrences matrix to generate S transducers to conduct topic spotting respectively. In this setup, we also use the same tf-idf re-weighting and pruning heuristics as in Subsection 5.5.1.4. As shown in Table 22, the semantic expansion for the term-document co-occurrence matrix used in LSA help to improve the topic spotting

Table 22: The effect of Semantic Expansion Using WordNet for the term-document co-occurrence matrix used in LSA on the topic spotting performance with LSRK

System/Method	N	K	Expansion	Accuracy
LSRK with LSA and tf-idf weighting	160K×2	750	No	53.56%
	160K×2	750	Yes	**55.33%**
	200K×2	750	No	53.33%
	200K×2	750	Yes	**54.67%**

performance by 1.77% and 1.34% in two different setups respectively.

5.5.1.6 LSRK using Probabilistic Topic Models

Probabilistic topic models make the bag-of-words assumption and both the parameters for the terms and the latent topics are constrained by the multinomial distributions respectively, (*i.e.*, summed up to 1). Since lower order n-gram features will definitely overlap with higher order observations, it is not trivial to consider and construct the higher order n-gram features beyond uni-grams with probabilistic topic models. One possibility is to build separate topic models for each order of n-grams, but with bi-grams or tri-grams, the issue of data sparsity becomes more precarious for the training of good topic models. Therefore in this setup, we only experiment with topic models built upon uni-gram features. The topic models we use for LSRK is learned from the same 2438 conversations text as in Subsection 5.5.1.4 using LDA. For the LDA estimation, we set the number of topics to 60 which is comparable to the total 66 topics among the 2438 conversations. The global Dirichlet prior parameter α is set to 1.0, and the topic distributions β is randomly initialized. When the LDA estimation procedure is done, we encode the estimated β into M transducers and employ the Fisher LSRK in two EM iterations as described in the Subsection 5.4.2.3 to conduct the topic spotting. As shown in Table 23, LSRK with the learned topic models can achieve the highest accuracy 62.67% on the Switchboard task in our setup, which is 5.11% further improvements over the best setup of LSRK with LSA.

Table 23: Classification Accuracies Comparison between N-grams Rational Kernels, LSRK with LSA and LSRK with Probabilistic Topic Models

System/Method	N	K	Accuracy
Unigram RK	-	-	27.33%
Bigram RK	-	-	28.22%
LSRK with LSA	1000K×2	750	57.56%
LSRK with Topic Models	-	60	**62.67%**

5.5.2 Experiments on HMIHY0300

Then we evaluated the LSRK framework for topic spotting on the AT&T HMIHY ("How May I Help You?") task. The data was recorded through HMIHY natural dialog system: all users are first prompted with the question "Hello, This AT&T. How May I Help You?", then the utterances are the recorded users' responses to this question (As the speech utterances in the HMIHY correspond to the direct users' responses to the prompted question, users' intentions are well embedded in those speech utterances. Thus we do not use the length threshold for the subset selection as in Switchboard.) The specific dataset we use for the LSRK evaluation is the HMIHY0300 initial collection, which contains 10,119 recorded speech utterances. All these utterances can be classified into 38 classes (call-types), eg., Billing Credit or Calling Plans. Note that some utterances may be assigned multiple classes. We first group those 10,119 utterances according to their call-type labels (for those utterances with multiple call-type labels, we just use the first label), then 10% of each utterances group will all go to the test set, the left 90% of each utterances group forms the training set. Finally, we have 997 utterances for testing and 9075 utterances for training (after filtering out some utterances with no transcriptions.)

5.5.2.1 The ASR system and WFSTs (lattices) Generation

The ASR system used for generating the WFSTs (lattices) for is very similar to what we described in Subsection 5.5.1.1. A tri-gram language model (LM) is trained on all transcriptions in the training set for decoding. The input features are also the MFCCs coupled with their linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT)

and feature-space maximum likelihood linear regression (fMLLR) for speaker adaptation during later iterations. The WER of the ASR system on the test set (997 utterances) is 26.23%. With this ASR system, we first trained a uni-gram LM using the all the transcriptions of the training set and then use it to generate lattices for all the training and test utterances. For the topic spotting experiments, we further selected those topics with more than 100 utterances. Finally, 7504 decoded lattices (WFSTs) are used for the SVM training and 842 WFSTs are used for the SVM testing.

5.5.2.2 Topic Spotting Using LSRK with LSA and tf-idf weighting

Since that one call-type of all the 38 classes is labeled as "OTHERS", we find the number of the utterances with this ambiguous label is fairly considerable (245 out of 842) in the test set. We believe this ambiguous class would obscure the true spotting performance difference, so we further exclude those utterance with label "OTHERS" in the calculation of the accuracy (This processing actually degrades the accuracy). The first setup for the LSRK evaluation is same as Subsection 5.5.1.4, and the baseline N-gram rational kernel system is same as Subsection 5.5.1.3.

We list the classification accuracy in terms of the different degrees of the pruning and the reduced dimensions we employs the SVD in the Table 24. As can be seen, the SVM with the N-gram rational kernels already can achieve the 79.56% and 75.71% accuracies compared to 27.33% and 28.22% we obtain on the Switchboard tasks. There should be two main explanations for the large performance difference on the two tasks: Firstly, speech utterances in the HMIHY tasks are uttered in a more intentional way considering the fact in the HMIHY dialog system after the system's prompts users should express their purposes of the calls directly. In the other hand, the ones in Switchboard are extracted from the spontaneous conversations after the topic is introduced. Secondly, the average length of the utterances in HMIHY is much longer than the ones in Switchboard which implies the decoded lattices will more likely contains meaningful words for the topic discrimination. With the LSRK with LSA and tf-idf weighting, we achieves from 2.01% to 2.85%

Table 24: Classification accuracies of LSRK with LSA and tf-idf wighting on the HMIHY0300, N is the number of non-diagonal elements left in $\mathcal{S}$ after pruning, K is the rank for the low dimensional term-document matrix approximation in LSA.

System/Method	N (pruning)	K (LSA)	Accuracy
Unigram RK	-	-	79.56%
Bigram RK	-	-	75.71%
LSRK using LSA and tf-idf weighting	40K×2	750	81.57%
	120K×2	750	81.74%
	160K×2	750	81.74%
	200K×2	750	81.74%
	40K×2	2000	80.90%
	40K×2	500	81.74%
	40K×2	250	**82.41%**

of the classification gains with different parameters. From the table, the performance gains from increasing the number of non-diagonal elements of the term-term similarity matrix appear to get saturated quickly which implies the relatively low number of the principle components is needed. This may be due to the sparsity of each document vector in the term-document matrix we used in the LSA (In the Switchboard experiments, we use the transcriptions of the whole conversations as the documents). Therefore, the highest spotting performance is achieved in the case of relatively low LSA dimensions with 250.

5.5.2.3 Topic Spotting Using LSRK with semantic expanded LSA and WordNet

Now let us look at the effect of the semantic expansion using WordNet in Table 25. In this case, the semantic expansion seems not helping that much: 0.67% improvements in one case and no improvements in the other case. Finally, we try using the term-term similarity matrix purely based on the WordNet and do the combination with tf-idf weighting as in Subsection 5.5.1.4, the accuracy is 81.90% which is 2.34% improvements over the N-gram baselines.

5.5.2.4 LSRK using Probabilistic Topic Models

Finally we do the experiments with LSRK using probabilistic topic models. The setup is very similar to the one described in Subsection 5.5.1.6. The only difference is we set

Table 25: The effect of Semantic Expansion Using WordNet for the term-document co-occurrence matrix used in LSA on the topic spotting performance with LSRK

System/Method	N	K	Expansion	Accuracy
LSRK with LSA and tf-idf weighting	40K×2	750	No	81.57%
	40K×2	750	Yes	**82.24%**
	120K×2	750	No	81.74%
	120K×2	750	Yes	81.74%
LSRK with WordNet	40K× 2	-	-	81.90%

Table 26: Classification Accuracies Comparison between N-grams Rational Kernels, LSRK with LSA and LSRK with Probabilistic Topic Models

System/Method	N	K	Accuracy
Unigram RK	-	-	79.56 %
Bigram RK	-	-	75.71%
LSRK with LSA	40K×2	250	82.41%
LSRK with Topic Models	-	60	**82.91%**

the number of topic to 30 when learning the LDA topic models from the transcriptions in the training set. As shown in Table 26, with LSRK using the learned topic models we can achieve the highest accuracy 82.91% which is 3.35% improvements over the N-gram baselines.

5.5.3 Discussions

In this section, we conduct experiments with LSRK under different setups and show it outperforms the N-gram rational kernel baselines on both Switchboard and HMIHY0300 tasks. From the experimental results, we can see that the advantage of the LSRK framework is more obvious when the task is more challenging where lots of filler and stop words are spontaneously uttered. Specifically, for the experiments using LSRK with LSA and tf-idf weighting, the spotting performance is monotonically enhanced with increasing N which is the number of non-diagonal elements left in the term-term similarity matrix after pruning as shown in both Table 21 and 24. However this effect seems much less obvious in the HMIHY experiments. This is because the utterances from the Switchboard data are

spoken in a more casual way compared to the ones from the HMIHY. Therefore, the increasing N will help in the sparseness issue of the spoken salient words in the Switchboard case but very limited in the HMIHY case. It also explains the reason why the semantic expansion using WordNet on the Switchboard also brings more spotting performance enhancement than the one in the HMIHY as shown in the Table 22 and 25. In Table 24, we fixed the parameter N to study the effect of the parameter K. In the HMIHY case, the spotting performance improves when we make lower rank approximation of the term-document matrix; however, we did not observe similar tendency in the Switchboard case. This is also largely due to the more spontaneous and ungrammatical speaking style of the Switchboard, which indicates that the co-occurrence patten between the spoken terms and the conversations in the Switchboard needs more dimensions than the one in the HMIHY case for proper semantic interpretations. In both experiments, the spotting performance of LSRK with probabilistic models is better than the one with LSA and tf-idf weighting as shown in Table 23 and 26. This result is consistent with the fact that probabilistic topic models have a more solid foundation in statistical inference compared to the algebra-based LSA models.

It should be noted that the ways to model the transducers M and S are not limited to what we have experimented in this section, one attractive feature of the LSRK framework is its capability to incorporate all available knowledge and techniques into one framework. It also worth noting that in all setups, learning the M and S transducers in the LSRK framework only needs text data in an unsupervised fashion. Therefore, the training data for learning the transducer M or S is not limited to the speech transcriptions with or without the topic labels. This is another appealing feature of the LSRK framework: much more training data become available for use to enhance the spotting performance. One of our future works for the LSRK framework is using large-scale text data to enhance the spotting performance further.

5.6 Summary

In this chapter, we propose the LSRK framework for topic spotting on conversational speech. The main idea is we map the WFSTs onto a reduced dimensional latent semantic space as in latent semantic analysis (LSA) or probabilistic topic models rather than mapping the WFSTs onto an n-gram feature space. A very appealing feature of the proposed LSRK framework is its capability to incorporate all available external knowledge and techniques into one unified WFST based framework to boost the topic spotting performance. In this regard, we generalize the LSRK using tf-idf weighting, LSA, WordNet and probabilistic topic models.

To validate the proposed LSRK framework, we evaluate it by conducting the SVM training and testing with it on two datasets, namely Switchboard and AT&T HMIHY0300. On both tasks, we comprehensively experiment several LSRK setups. The experimental results show that with the proposed LSRK framework, we can achieve significant and consistent spotting performance gains over the N-gram baseline setup.It should be noted that the ways to use LSRK is not limited to what we have experimented, in practice, certain form of the combinations can be flexibly integrated into the LSRK framework. It is also worth noting that learning a LSRK only needs text data in an unsupervised way, which makes more text training data becoming available in the LSRK framework.

CHAPTER 6

NON-UNIFORM DISCRIMINATIVE TRAINING WITH LATENT SEMANTIC RATIONAL KERNELS

6.1 Introduction

In Chapter 2-4, we mainly discuss how to build more robust acoustic models against the challenging speaking styles and acoustical environments for the conversational speech recognition and understanding. In Chapter 5, we focus on how to make a robust semantic decision given the speech observations and an ASR system. A natural question to ask is that whether we can combine all the presented techniques together towards a more robust speech recognition and understanding system, *i.e.*, propose a way to build acoustic models which will lead to more robust semantic decisions rather than higher word recognition accuracies in challenging conditions.

Whereas most of the ASR systems focus on the minimization of WER, the reduction of WER does not necessarily lead to a better speech understanding system. It is obvious that some keywords or salient words will carry more important semantic information but the system which uses the WER as the performance metric treats the errors of each individual word uniformly. A good example is presented in Chapter 2, where the system with the lowest WER commits more errors with respect to those keywords compared with the system trained with the non-uniform MCE. Therefore, it is clearly important to formulate a new criterion for acoustic modeling not only for speech recognition but also for speech understanding.

In this chapter, based on the latent semantic rational kernels (LSRKs) framework we proposed in Chapter 5 and following the same spirit of non-uniform discriminative training (DT) [41, 42, 43, 44], we propose non-uniform DT of deep neural networks (DNNs) with LSRKs. Sequential discriminatively trained DNNs systems [87, 135, 88] have shown

consistent gains over frame-level cross-entropy trained DNN system. Moreover, sequential DT of DNNs allows us to design a new objective function that directly links to the LSRKs framework. Recently the word representations learned by the neural networks [136, 137, 138] have been shown to capture the syntactic and semantic regularities in language and these word representations can be efficiently learned on a very large text corpus. Therefore, we also present a way to incorporate these neural network learned word representations into our LSRK framework.

The remainder of this chapter is organized as follows. We review the LSRK framework and the neural network learned latent semantic representations, and then present a way to incorporate the learned representations into the LSRK framework in Section 6.2. In Section 6.3, we formulate a new objective function for non-uniform DT of DNNs with LSRKs and elaborate how it can be implemented efficiently. We report our experimental results in Section 6.4 and conclude this chapter in Section 6.5.

6.2 Neural Network Learned Latent Semantic Representations and Rational Kernels

In the LSRK framework, the input speech lattice is first represented by a vector in a certain latent semantic space. Therefore, the representation learning for each word or document is obviously crucial to the LSRKs. In Chapter 5, we have generalized the LSRKs using multiple types of the document representations. However, these representation learning techniques may not be efficiently applied when the training text corpus is very large. To make the LSRK framework scalable to a very large text training corpus, we further generalize LSRKs using neural network learned representations.

6.2.1 Neural Network Learned Latent Semantic Representations

Neural network learned latent semantic representations are closely related to the neural network based language models (LMs) [21] [22]. Two basic architectures of neural network LMs are shown in Fig.23. In the feedforward architecture, the word history $\mathbf{w}(t-3)$, $\mathbf{w}(t-2)$,

$\mathbf{w}(t-1)$ are fed into a shared weight matrix W_{ih}. Note that each $\mathbf{w}(t)$ is a vector of which the number of dimensions is the same as the vocabulary size V and that the only non-zero dimension corresponds to the index of the word. After the sigmoid non-linearity, the hidden vector is given by a concatenation of the three corresponding hidden vectors,

$$\mathbf{h}(t) = \left(\mathbf{h}_1^T(t), \mathbf{h}_2^T(t), \mathbf{h}_3^T(t)\right)^T, \tag{129}$$

where $\mathbf{h}_i(t) = \text{sigmoid}(W_{ih}\mathbf{w}(t-i) + \mathbf{b}_{ih})$. Finally the output vector $w(t)$ is given by,

$$\mathbf{w}(t) = \text{softmax}(W_{ho}\mathbf{h}(t)). \tag{130}$$

To capture the long context dependency, the recurrent architecture was introduced into neural network LMs in [22]. As shown in Fig.23 (b), the hidden activations recursively need the previous values of the hidden units,

$$\mathbf{h}(t) = \text{sigmoid}\left(W_{ih}\mathbf{w}(t-1) + \mathbf{b}_{ih} + W_{hh}\mathbf{h}(t-1) + \mathbf{b}_{hh}\right), \tag{131}$$

then the output activations are given by,

$$\mathbf{w}(t) = \text{softmax}(W_{ho}\mathbf{h}(t)). \tag{132}$$

The training of the parameters, *i.e.*, $W_{ih}, W_{ho}, \mathbf{b}_{ih}$, is typically done using SGD. And the parameters of recurrent hidden layer, *i.e.*, $W_{hh}, \mathbf{b}_{hh}$, need the BPTT to accumulate the necessary gradients.

When the recurrent architecture is used, it was found in [136] that if we take a certain column of W_{ih} as the vector representation for the corresponding word, this representation is good at capturing syntactic and semantic relationships in language, *e.g.*, the male/female relationship can be automatically learned, and in this vector representation space, "King - Man + Woman" will result in a vector very close to "Queen". However, although some strategies are introduced in [139] to speed up the RNNLM training, the recurrent architecture prevents us from making further accelerations via parallel computing. Therefore, more efficient word representation learning architectures are introduced in [137] and [138]. As

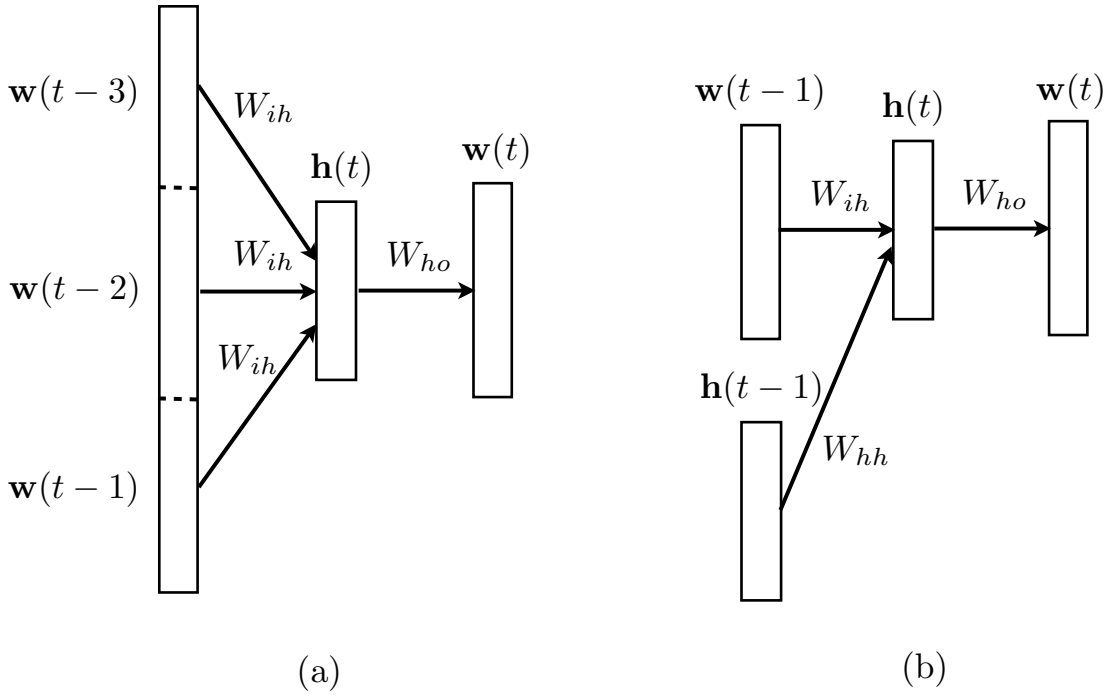

Figure 23: The architecture of neural network based language models: (a) the feedforward neural network LM, note that W_{ih} is shared across multiple words (b): the recurrent neural network LMs

shown in Fig.24, there are two kinds of architectures, continuous bag-of-words (CBOW) and skip-gram models. They are very similar to the feedforward neural network LMs, the only difference is that there is no sigmoid non-linearity in producing the hidden vector $\mathbf{h}(t)$. For more efficient learning, the hidden vectors $h(t)$ is simply the averaged vector of multiple input word features,

$$\mathbf{h}(t) = \frac{1}{2\tau} \sum_{i=-\tau, i\neq 0}^{\tau} W_{ih}\mathbf{w}(t-i). \tag{133}$$

With these simple architectures, the training process can be dramatically accelerated using some distributed optimization framework, *e.g.*, Downpour SGD [140]. Thus the word representations can be learned efficiently on very large text corpus.

6.2.2 LSRKs Using Neural Network Learned Representations

There are two types of formulations in the LSRK framework,

$$k_n(A_1, A_2) = \begin{cases} w[(A_1 \circ T) \circ M \circ M^{-1} \circ (T^{-1} \circ A_2)] \\ w[(A_1 \circ T) \circ S \circ (T^{-1} \circ A_2)] \end{cases}. \tag{134}$$

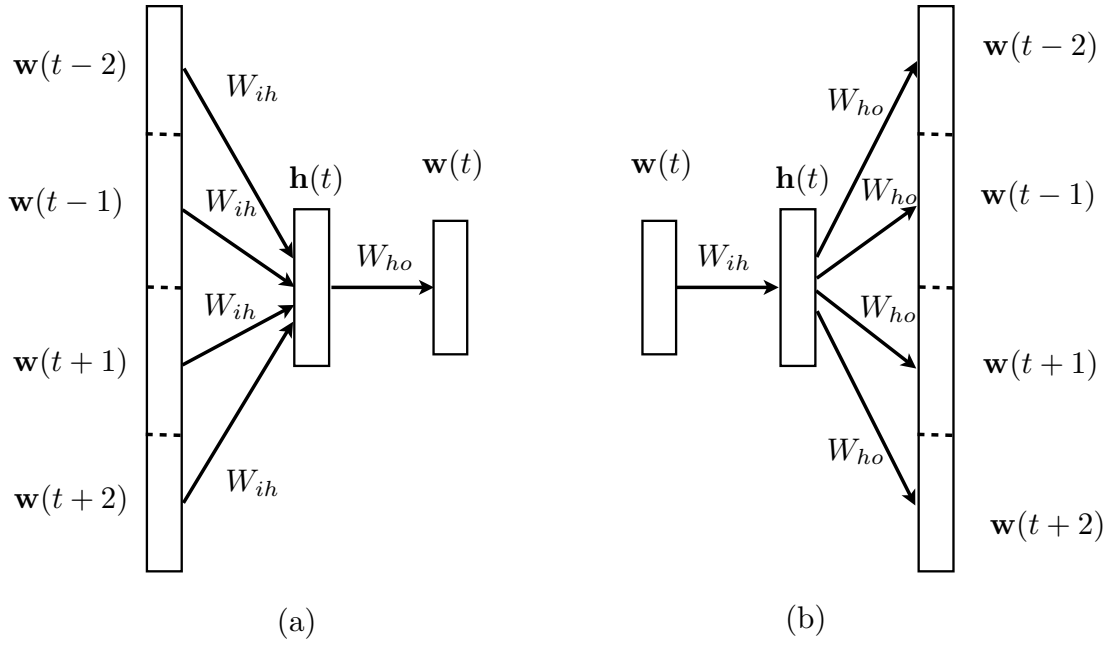

Figure 24: More efficient word representation learning architecture: (a) CBOW: no sigmoid non-linearity, $\mathbf{h}(t)$ is the averaged vector rather than the concatenation, note that W_{ih} is shared across multiple input words (b)Skip-gram: the reversed CBOW structure, note that W_{ho} is shared across multiple output words

The first one is based on the transducer M which encoding the projection matrix which transforms certain WFST n-gram features into certain low dimensional representations; the second is based on the transducer S which encodes the term-term similarity information. Since we make BOW assumptions in the LSRK framework, LSRKs using neural network learned representations are very similar to the generalizations using vector space models as discussed in Chapter 5.

Since each column of the weight matrix W_{ih} corresponds to the neural network learned vector representation $\mathbf{d}(w)$ for the word w, the projection matrix used for mapping the original vector into the latent semantic space is then exactly the weight matrix W_{ih}. We first define a transducer M_{NN} which encodes the projection matrix which contains two states and $V \times H$ arcs between them (H is the dimension of the hidden units when training word representations). The input labels of each arc represent words indices and output labels represent the dimensional indices of the hidden units. The weight on the arc with input and

output labels pair (i, h) will be $W_{ih}(i, h)$. Therefore the LSRK with neural network learned word representations can be written as,

$$\begin{aligned} K(A_1, A_2) &= w\left[A_1 \circ T \circ M_{\text{NN}} \circ M_{\text{NN}}^{-1} \circ T^{-1} \circ A_2\right] \\ &= w\left[A_1 \circ T \circ S_{\text{NN}} \circ T^{-1} \circ A_2\right]. \end{aligned} \tag{135}$$

Another form of the LSRK with neural network is treating $M_{\text{NN}} \circ M_{\text{NN}}^{-1}$ as one transducer S_{NN}. This form is more appealing in terms of its term-term similarity suggesting that the LSRK with neural network learned word representations can also be easily combined with other schemes, *e.g.*, tf-idf weighting or WordNet.

6.3 Non-uniform Discriminative Training of Deep Neural Networks with Latent Semantic Rational Kernels

6.3.1 Sequential Discriminative Training of DNNs

The most commonly used DNN training criterion for speech recognition is the frame level cross-entropy. Let $\{X\}_{r=1}^{R}$ be the whole training set which contains R speech utterances, the negative cross-entropy loss objective function is given by,

$$\mathcal{L}(\theta) = -\sum_{r=1}^{R}\sum_{t=1}^{T_r}\sum_{j=1}^{J} \boldsymbol{\delta}_t(j) \log \mathbf{y}_t(j), \tag{136}$$

where $\boldsymbol{\delta}_t(j)$ is the j^{th} element of the label vector at frame t and $\mathbf{y}_t$ is the output vector from the DNN. Note that no matter what form of the loss function is used in the DNN training, the only difference reflected in the backpropagation lies in the error signals we backpropagate to the previous hidden layers and taking partial derivatives of the loss objective function with respect to the pre-nonlinearity activations of output layer will generate the error signals we need. For the negative cross-entropy loss, the error vector to be backpropagated to the DNN is given by,

$$\boldsymbol{\epsilon}_t(j) = \mathbf{y}_t(j) - \boldsymbol{\delta}_t(j). \tag{137}$$

However, the hybrid DNN-HMM systems trained with utterance level discriminative training criteria which have more direct links with the WER metric used in the ASR system,

e.g. boosted MMI [27], MCE [25], sMBR [87, 135] and MPE [26], have shown consistent improvements [88] over the frame-level cross-entropy trained systems. As mentioned earlier, a DNN-HMM hybrid system uses pseudo log-likelihood as the state emissions,

$$\log p(\mathbf{x}_t|s_j) \propto \log p(s_j|\mathbf{x}_t) - \log p(s_j), \tag{138}$$

where the state priors $\log p(s_j)$ are independent of the parameters of the network, which are estimated using the state alignments on the training speech data. The outputs from the DNNs with the last softmax layer represent the posteriors $p(s_j|\mathbf{x}_t)$ so the pre-nonlinearity activations of output layer represent $\log p(s_j|\mathbf{x}_t)$. Therefore, the error signals we need are obtained using the equation,

$$\frac{\partial \mathcal{L}(\theta)}{\partial \log p(s_j|\mathbf{x}_t)} \propto \frac{\partial \mathcal{L}(\theta)}{\partial \log p(\mathbf{x}_t|s_j)}. \tag{139}$$

For MMI or boosted MMI, the loss objective function is,

$$\mathcal{L}_{\text{MMI}}(\theta) = -\sum_{r=1}^{R} \log \frac{P(X_r|W_r)P(W_r)}{\sum_W P(X_r|W)P(W)e^{-b\mathcal{A}(W,W_r)}}, \tag{140}$$

where $\mathcal{A}(W, W_r)$ is the accuracy function which specifies the number of correct events in the hypothesis W corresponding to different granularity of labels. The error signal we need is given by,

$$\frac{\partial \mathcal{L}_{\text{MMI}}(\theta)}{\partial \log p(\mathbf{x}_t|s_j)} = \gamma_j^{\text{den}}(t) - \gamma_j^{\text{num}}(t), \tag{141}$$

where $\gamma_j^{\text{num}}(t)$, $\gamma_j^{\text{den}}(t)$ are occupancies derived from the labels and lattices respectively. For the MPE/sMBR loss objective functions,

$$\mathcal{L}_{\text{MPE}}(\theta) = -\sum_{r=1}^{R} \sum_{W} \mathcal{A}(W_r, W) \frac{P(X_r|W)P(W)}{\sum_{W'} P(X_r|W')P(W')}. \tag{142}$$

The error signal we need is given by,

$$\frac{\partial \mathcal{L}_{\text{MPE}}(\theta)}{\partial \log p(\mathbf{x}_t|s_j)} = \gamma_j^{\text{den}}(t) \left(\overline{\mathcal{A}}(W, W_r) - \overline{\mathcal{A}}(W(t) = s_j, W_r) \right), \tag{143}$$

where $\{W|W(t) = s_j\}$ is the hypothesized word sequences set which contains state s_j at frame t and $\overline{\mathcal{A}}(W_r, W)$ is the averaged accuracy between W_r and all hypothesized W.

6.3.2 Non-uniform Discriminative Training of DNNs with LSRKs

The advantage of the sequential DT on DNNs is that one can flexibly design a new objective function that directly links to the system evaluation metric. As in Section 6.3.1, all the objective functions, *e.g.*, MMI, sMBR and MPE, are to minimize the WER of the system. However, the minimization of WER does not necessarily leads to the error minimization in the LSRK framework. Thus our task is to formulate a new objective function for the DT of DNNs and derive the form of the error signals that we will backpropagate to the DNNs. Denote by $\mathbf{d}(W)$ the latent semantic representation for the word sequence W. LSRKs defined on the two input speech utterances X_1 and X_2 can be viewed as the inner product between two corresponding *expected* representations over their respective word sequence posteriors $P(W|X)$,

$$K(X_1, X_2) = \langle \mathbb{E}_{P(W|X_1)}[\mathbf{d}(W)], \mathbb{E}_{P(W|X_2)}[\mathbf{d}(W)] \rangle \,. \tag{144}$$

Then the kernels will be used in the SVM to classify the speech utterance. Suppose W_1 and W_2 are label transcriptions of X_1 and X_2. Ideally the value of the LSRK between X_1 and X_2 should be equal to,

$$K(X_1, X_2) = \langle \mathbf{d}(W_1), \mathbf{d}(W_2) \rangle \,. \tag{145}$$

Therefore, to minimize the errors in the LSRK framework, we need to minimize the distance between the representations for label transcriptions W_r and the *expected* representations over the posterior distribution $P(W|X_r)$,

$$\theta^* = \arg\min_{\theta} \text{Distance} \left(\mathbf{d}(W_r), \mathbb{E}_{P(W|X_r)}[\mathbf{d}(W)] \right) . \tag{146}$$

Note that,

$$P(W|X_r) = \frac{P(X_r|W)P(W)}{\sum_{W'} P(X_r|W')P(W')}, \tag{147}$$

we are trying to train a set of acoustic model parameters $P(X|W)$ to satisfy (146). Regarding the distance metric between two representations, for an efficient implementation (mainly for the forward-backward dynamic programming purpose), we use the negated inner product. Given a set of training utterances, $\{X_r\}_{r=1}^{R}$, we define the objective function of

non-uniform discriminative training with LSRKs as,

$$\mathcal{L}(\theta) = -\sum_{r=1}^{R} \langle \mathbf{d}(W_r), \mathbb{E}_{P(W|X_r)}[\mathbf{d}(W)] \rangle . \tag{148}$$

With the objective function defined above, we need to derive the error signal, *i.e.,* $\frac{\partial \mathcal{L}(\theta)}{\partial \log p(\mathbf{x}_t|s_j)}$, for the backpropagation on the DNNs. For the derivation, the objective function needs to be rewritten as,

$$\begin{aligned}
\mathcal{L}(\theta) &= -\sum_{r=1}^{R} \langle \mathbf{d}(W_r), \mathbb{E}_{P(W|X_r)}[\mathbf{d}(W)] \rangle \\
&= -\sum_{r=1}^{R} \mathbb{E}_{P(W|X_r)} \langle \mathbf{d}(W_r), \mathbf{d}(W) \rangle \\
&= -\sum_{r=1}^{R} \sum_{W} \langle \mathbf{d}(W_r), \mathbf{d}(W) \rangle P(W|X_r) \\
&= -\sum_{r=1}^{R} \sum_{W} \langle \mathbf{d}(W_r), \mathbf{d}(W) \rangle \frac{P(X_r|W)P(W)}{\sum_{W'} P(X_r|W')P(W')}.
\end{aligned} \tag{149}$$

The form of the objective function now is very similar to the one of sMBR or MPE except that the accuracy function $\mathcal{A}(W_r, W)$ changes to $\langle \mathbf{d}(W_r), \mathbf{d}(W) \rangle$. Therefore we have,

$$\begin{aligned}
\frac{\partial \mathcal{L}(\theta)}{\partial \log p(\mathbf{x}_t|s_j)} &= \gamma_j^{\text{den}}(t) \left(\mathbb{E}_{P(W|X_r)} \langle \mathbf{d}(W_r), \mathbf{d}(W) \rangle \right. \\
&\quad \left. - \mathbb{E}_{P(W(t)=s_j|X_r)} \langle \mathbf{d}(W_r), \mathbf{d}(W) \rangle \right) \\
&= \gamma_j^{\text{den}}(t) \left(\langle \mathbf{d}(W_r), \mathbb{E}_{P(W|X_r)}[\mathbf{d}(W)] \rangle \right. \\
&\quad \left. - \left\langle \mathbf{d}(W_r), \mathbb{E}_{P(W(t)=s_j|X_r)}[\mathbf{d}(W)] \right\rangle \right),
\end{aligned} \tag{150}$$

where $\{W|W(t) = s_j\}$ is the hypothesized word sequences set which contains state s_j at frame t. $\gamma_j^{\text{den}}(t)$ is the occupancy probability for state s_j at frame t, *i.e.*, $P(s_j|X_r)$. So the error signal we will backpropagate to the DNNs is actually the difference between two inner products weighted by the state posteriors where the first inner product is between the label transcription representation and the expected representation over all the word hypothesis, and the second is between the label transcription representation and expected one over those hypothesized word sequences contain state s_j at frame t. To efficiently accumulate

the error signals $\frac{\partial \mathcal{L}(\theta)}{\partial \log p(\mathbf{x}_t|s_j)}$, a similar forward-backward dynamic programming strategy as in MPE/MWE can be used where the local accuracy function at frame t in the WFST lattice is defined as,

$$\mathcal{A}(w_r(t), w(t)) = \langle \mathbf{d}(w_r(t)), \mathbf{d}(w(t)) \rangle \,, \tag{151}$$

here $w_r(t)$ and $w(t)$ is the reference word and the hypothesized word at frame t. Note that the use of the local accuracy function is based on the assumption when evaluating $\langle \mathbf{d}(W_r), \mathbf{d}(W) \rangle$ we only take into accounts the inner product components corresponding to the time-aligned reference and hypothesized word pairs. This is reasonable because $\langle \mathbf{d}(W_r), \mathbf{d}(W) \rangle$ is purely based on the bag-of-words assumption which fully ignores the order of the words, while we obviously need to consider the order of words when dealing with the observations for the acoustic modeling.

6.4 Experiments

In this section, we report the topic spotting results for sequential discriminatively trained DNNs with LSRKs using neural network learned representations on the subset of Switchboard.

6.4.1 Neural Network Learned Word Representations

To establish the LSRKs with neutal network learned representations, we first train a neural network using the word2vec toolkit [137]. The training text corpus we use is the first billion bytes of the English Wikipedia [1]. We use the CBOW architecture as described in Section 6.2.1 with 200 hidden dimensions and 5-words windows. After the network is trained, we take the corresponding column vectors as the word representations for each word in the vocabulary. Note that the vocabulary is no longer the same as Switchboard's. To examine the learned word representations, we find the top similar words according to cosine similarity with the topics we used in the topic spotting experiment and listed some examples in the Table 27 (if multiple words in a topic, we use the averaged vectors).

[1] available on `http://mattmahoney.net/dc/textdata.html`

Table 27: Top 10 Relevant Words to 5 Selected Topics Used in the Topic Spotting Experiment according to the neural network learned word representations on Wikipedia text

Topic Words	Top 10 Relevant Words
Air Pollution	pollutants, transboundary, groundwater, conditioner, breathable, emissions, smog, siltation, contrails, ozone
Buying a Car	cars, truck, subcompact, luxury, dealerships, suv, minivans, dealers, convertible, automobiles
Drug Testing	drugs, medication, treatment, therapeutic, antiretroviral, tests,mefloquine,medications,therapy,diagnostic
Pets	raccoons, animals, ferrets, adoptable, capybaras, dogs, chickens, chameleons, chihuahuas, goannas
Women's Role	child, roles, girl, man, lover, pregnant, prostitute, herself, young, mother

6.4.2 Sequential DTs of DNNs with LSRKs

We build several DNN-HMM systems to generate the lattices for topic spotting using LSRKs. To get the initial state alignments, we use the built GMM-HMM system described in Section 2.5.1. For the DNN-HMM systems, we first do generative pre-training using RBMs, and stack them together in the end to initialize the DNN with 7 2048-dim hidden layers. Note that the training data used in the DNN-HMM system is almost doubled to approximately 200 hours. The features fed into DNN-HMM systems are MFCC features coupled with their LDA/MLLTs and speaker adapted transforms estimated using fMLLRs. The first DNN-HMM system, which we will refer to as DNN-HMM CE, is trained using cross-entropy criteria. We use 256 minibatch and 0.008 as the initial learning rate. The learning rate scheduling mechanism is the same as described in Chapter 3 and 4.

Then we use the DNN-HMM CE system to realign the training data and generate the lattices for the sequential DTs of DNNs. Note that these lattices are generated using unigram LMs which are used in the DT, not the lattices used in the topic spotting experiments. With the new generated state alignments and lattices, we first conduct the conventional DT training on DNNs using sMBR and MWE, which we will refer to as DNN-HMM sMBR and DNN-HMM MWE. Finally, with the word representations learned in the last section,

Table 28: WERs of different Sequential Discriminatively Trained DNN-HMM systems on HUB5 English evaluation set

Systems	LM scales	WERs
DNN-HMM CE	10	23.2%
DNN-HMM sMBR	14	**21.6%**
DNN-HMM MWE	14	21.7%
DNN-HMM MLSE	13	21.7%

we conduct the proposed non-uniform DT with LSRKs, which we will refer to as the DNN-HMM MLSE system.

With all trained hybrid DNN-HMM systems and the tri-gram language models, we first do the ASR experiments on the standard HUB5 evaluation set. The WER results are listed in Table 28. As can be seen, the results show there are no obvious differences between different discriminatively trained DNN-HMM systems while all the systems achieve significant WER gains over the cross-entropy trained system. The DNN-HMM sMBR obtain the lowest 21.6% WER.

6.4.3 Topic Spotting on the Switchboard Subset

With all discriminatively trained hybrid DNN-HMM systems, we conduct the topic spotting experiments on a subset of Switchboard. The subset we use here is the same as the one used in Chapter 5, please find more details on the selection of the subset in Section 5.5.1.2. With a uni-gram LM trained on the whole transcriptions of the dataset, we use four DNN-HMM ASR systems trained in last section to generate the lattices for each utterance in the selected subset respectively.

All the generated lattices will be the inputs fed into the LSRK framework with the neural network learned representations as described in Section 6.4.1. Note that since we train word representations on a held-out text corpus, to overcome the issue of OOVs, we simply use all-zero vectors for those OOVs. We use the second form of the LSRK formulation in (134). Although the second form incurs more computations compared to the first one, the negative values in some dimensions of word representations will cause problems within the

Table 29: Topic classification accuracies of LSRKs with neural network learned representations on the lattices generated from different sequential discriminatively trained DNN-HMM systems on Switchboard Subset.

Systems	N (pruning)	Accuracies
NN LSRK + DNN CE	80K×2	52.22%
NN LSRK + DNN sMBR	80K×2	53.56%
NN LSRK + DNN MWE	80K×2	54%
NN LSRK + DNN MLSE	80K×2	**55.11%**

log-semiring of the WFST framework which prevents us from using the first form. Using the same pruning scheme described in Section 5.5.1.4, we can significantly reduced the computational complexity.

Table 29 shows the topic classification accuracies of the four different systems. We keep 160K most significant non-diagonal elements in the S matrix for LSRK. Although the DNN-HMM sMBR system achieves the lowest WER, the system trained using the proposed non-uniform DT with LSRK obtains the highest 55.11% topic classification accuracy. This illustrates the ASR system commits the lowest WER does not necessarily lead to the best system for the semantic decoder. And our proposed method fills this gap by directly linking the information of the semantic decoder to the DT objective function.

6.5 Summary

In this chapter, based on the latent semantic decoder (LSRKs) proposed in Chapter 5 and following the same spirit of non-uniform DT in Chapter 2 we propose the non-uniform DT method of DNNs with LSRKs which directly links the information of the semantic decoder to the DT objective function. We also present one way to incorporate the neural network learned word representations which can be efficiently learned on a very large text corpus into LSRKs.

We validate the proposed methods using 200 hours of speech training data to train the DNN-HMM acoustic models and using a subset of the Switchboard database to train and test the semantic decoders. Experimental results show the proposed method can lead the

acoustic modeling to the best system with respect to the semantic decoder.

CHAPTER 7

CONCLUSION

7.1 Summary and Contributions

In this dissertation, we have proposed five methods/systems towards a more robust unconstrained conversational speech recognition and understanding system. Fig.25 shows an overview of all proposed methods where I-III aim to build more robust acoustic models, IV performs as a robust semantic decoder working with an ASR system and V bridges the two components by using the information in IV as the feedbacks to re-train the acoustic models.

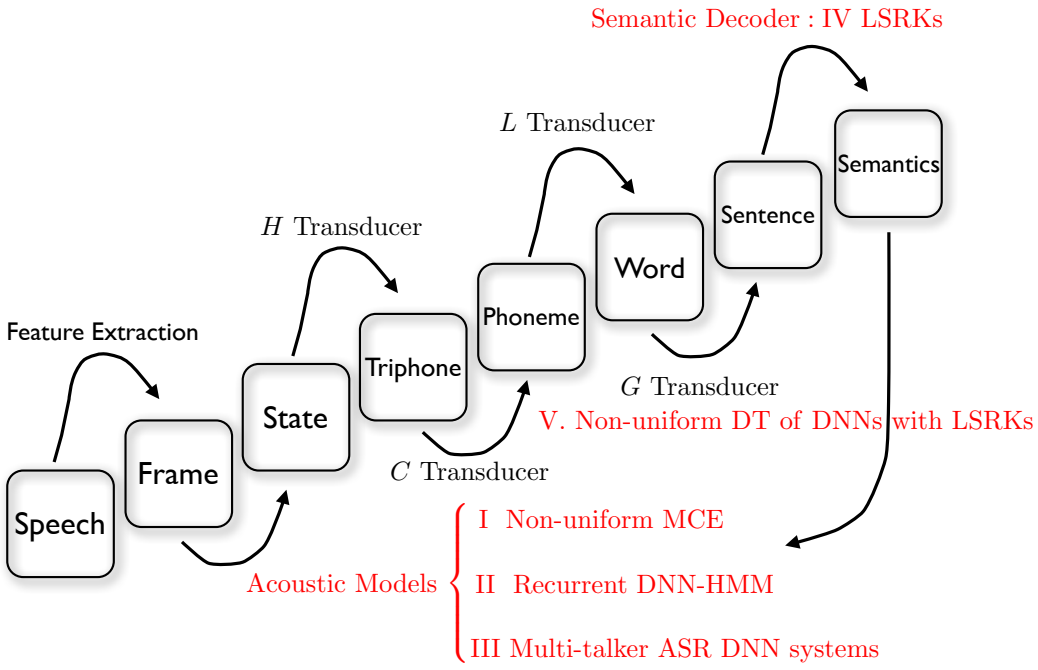

Figure 25: An Overview of the Proposed Methods and System

The contributions of this dissertation are summarized as follows:

I A non-uniform minimum classification error (MCE) approach is proposed which can achieve consistent and significant keyword spotting performance gains on both English and Mandarin large-scale spontaneous conversational speech tasks (Switchboard and HKUST Mandarin CTS).

II A hybrid recurrent DNN-HMM system is proposed for robust acoustic modeling and a new way of backpropagation through time (BPTT) is introduced. The proposed system achieves state-of-the-art performances on two benchmark datasets, the 2nd CHiME challenge (track 2) and Aurora-4, without front-end preprocessing, speaker adaptive training or multiple decoding passes.

III To study the specific case of conversational speech recognition in the presence of competing talkers, several multi-style training setups of DNNs are investigated and a joint decoder operating on multi-talker speech is introduced. The proposed combined system improves upon the previous state-of-the-art IBM superhuman system by 2.8% absolute on the 2006 speech separation challenge dataset.

IV Latent semantic rational kernels (LSRKs) are proposed for spotting the semantic notions on conversational speech. The proposed framework is generalized using tf-idf weighting, latent semantic analysis, WordNet, probabilistic topic models and neural network learned representations and is shown to achieve substantial topic spotting performance gains on two conversational speech tasks, Switchboard and AT&T HMIHY initial collection.

V Non-uniform sequential DT of DNNs with LSRKs is proposed which directly links the information of the semantic decoder to the objective function of the DT. The experimental results on the subset of Switchboard show the proposed method can lead the acoustic modeling to a more robust system with respect to the semantic decoder.

7.2 Future Perspectives

One remaining problem of the proposed non-uniform MCE training method is how to rapidly adapt the trained models to a new keyword set. In the model space, the only way is to retrain the models based on the new keyword set requiring the system to keep multiple sets of models when dealing with various keywords sets. This will lead to an unwieldy

decoder with unreasonable latencies since it needs to reload the models when switching between different keyword sets. Instead, feature space DT [141] methods will enable us to train the feature transform for each keyword set and keep only one set of back-end models and decoder.

A natural extension of the proposed recurrent DNN-HMM system is the sequential DT of recurrent DNN. The sequential criteria are much more computationally demanding than the frame level cross-entropy and the training time on the recurrent DNN is usually 4-5 times slower than the feedforward one in our experiments. Therefore, the main issue here is how to speed up the training process via parallel computing. So far there are mainly two types of the distributed optimization systems for DNNs. The first one is based on the second-order optimization methods, *e.g.*, Hessian Free for sMBR [142, 135], which usually use the overall training data for the estimation of the gradients and subsamples for stochastic estimation of the curvature. The other one is based on asynchronous SGD, *e.g.*, HogWild [143] and Distbelief [140]. The bottleneck of training on the proposed recurrent DNN is that the forward propagation has to be conducted frame by frame. The use of large batch of the second-order optimization is more appealing for the recurrent architecture since the large size of the batch will allow the propagation to be parallelized over multiple speech utterances.

In the multi-talker ASR scenario, how to track one of the multiple speakers is very important. As discussed in the end of Chapter 4, although the DNN-based unsupervised mask learning approach we have tried has not worked yet, we believe a good initialization would improve the estimation as shown in [107]. Furthermore, informed by a well-estimated mask for the target speaker, the complexity of the joint decoder can be largely reduced making the system scalable to an LVCSR task.

REFERENCES

[1] T. T. Kristjansson, J. R. Hershey, P. A. Olsen, S. J. Rennie, and R. A. Gopinath, "Super-human multi-talker speech recognition: the IBM 2006 speech separation challenge system.," in *INTERSPEECH*, ISCA, 2006.

[2] T. Virtanen, "Speech recognition using factorial hidden Markov models for separation in the feature space.," in *INTERSPEECH*, ISCA, 2006.

[3] J. Godfrey, E. Holliman, and J. McDaniel, "Switchboard: telephone speech corpus for research and development," in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, vol. 1, pp. 517–520 vol.1, 1992.

[4] D. Pallett, "National institute of science and technology," *Proceedings of the IEEE*, Feb 1999.

[5] B.-H. Juang and S. Furui, "Automatic recognition and understanding of spoken language - a first step toward natural human-machine communication," *Proceedings of the IEEE*, vol. 88, pp. 1142–1165, Aug 2000.

[6] L. Deng and X. Huang, "Challenges in adopting speech recognition," *Commun. ACM*, vol. 47, pp. 69–75, Jan. 2004.

[7] B.-H. Juang, "Speech recognition in adverse environments," *Computer Speech and Language*, vol. 5, 1992.

[8] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, pp. 357–366, Aug 1980.

[9] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Am.*, vol. 57, pp. 1738–52, Apr. 1990.

[10] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, Feb 1989.

[11] B. H. Juang and L. R. Rabiner, "Hidden markov models for speech recognition," *Technometrics*, vol. 33, pp. 251–272, Aug. 1991.

[12] S. J. Young, J. J. Odell, and P. C. Woodland, "Tree-based state tying for high accuracy modelling," in *HLT*, 1994.

[13] F. Jelinek and R. L. Mercer, "Interpolated estimation of Markov source parameters from sparse data," in *Proceedings, Workshop on Pattern Recognition in Practice* (E. S. Gelsema and L. N. Kanal, eds.), pp. 381–397, Amsterdam: North Holland, 1980.

[14] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 35, pp. 400–401, Mar 1987.

[15] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1990.

[16] R. Kneser and H. Ney, "Improved backing-off for M-gram language modeling," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1, pp. 181–184 vol.1, May 1995.

[17] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, (Stroudsburg, PA, USA), pp. 310–318, Association for Computational Linguistics, 1996.

[18] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra, "A maximum entropy approach to natural language processing," *Comput. Linguist.*, vol. 22, pp. 39–71, Mar. 1996.

[19] R. Rosenfeld, *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, School of CS, CMU, 1994.

[20] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer, 2nd ed., 2006.

[21] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

[22] T. Mikolov, *Statistical Language Models Based on Neural Networks*. PhD thesis, BRNO University of Technology, 2012.

[23] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 20, no. 1, pp. 69–88, 2002.

[24] L. Bahl, P. Brown, P. de Souza, and R. Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," in *Proc. ICASSP1986*, pp. 49–52, 1986.

[25] B.-H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," *IEEE Trans. Signal Process.*, vol. 40, pp. 3043–3054, Dec. 1992.

[26] D. Povey, *Discriminative learning for large vocabulary speech recognition*. PhD thesis, Univ. of Cambridge, 2004.

[27] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted mmi for model and feature space discriminative training," in *Proc. ICASSP2008*, pp. 4057–4060, 2008.

[28] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional hmm systems," in *PROC. ICASSP*, pp. 1635–1638, IEEE, 2000.

[29] F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky, "Probabilistic and bottle-neck features for LVCSR of meetings," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, pp. IV–757–IV–760, April 2007.

[30] H. A. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Norwell, MA, USA: Kluwer Academic Publishers, 1993.

[31] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[32] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, *Deep Neural Network Language Models*. Montréal, Canada: Association for Computational Linguistics, June 2012.

[33] B.-H. Juang and S. Furui, "Automatic recognition and understanding of spoken language – a first step towards natural human-machine communication," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1142–1165, 2000.

[34] B.-H. Juang, "From speech recognition to understanding: Shifting paradigm to achieve natural human-machine communication," in *Proc. 16th ICA and 135th Meeting ASA*, pp. 617–618, 1998.

[35] A. L. Gorin, G. Riccardi, and J. H. Wright, "How may i help you?," *Speech Communication*, vol. 23, pp. 113–127, 1997.

[36] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Speech Audio Process.*, vol. 26, no. 1, pp. 43–49, 1978.

[37] J. G. Wilpon and D. B. Roe, "AT&T telephone network applications of speech recognition," in *Proc. COST232 Workshop*, 1992.

[38] M. Rahim, C. Lee, and B.-H. Juang, "Discriminative utterance verification for connected digits recognition," *IEEE Transactions on Speech and Audio Processing*, pp. 266–277, 1997.

[39] J. Wilpon, L. Rabiner, C.-H. Lee, and E. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden Markov models," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, pp. 1870–1878, 1990.

[40] R. C. Rose, "Keyword detection in conversational speech utterances using hidden Markov model based continuous speech recognition," *Computer Speech and Language*, vol. 9, pp. 309–333, 1995.

[41] Q. Fu, Y. Zhao, and B.-H. Juang, "Automatic speech recognition based on non-uniform error criteria," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 3, pp. 780–793, 2012.

[42] C. Weng and B.-H. Juang, "Recent development of discriminative training using non-uniform criteria for cross-level acoustic modeling," in *Proc. ICASSP2011*, pp. 5332–5335, 2011.

[43] C. Weng and B.-H. Juang, "A comparative study of discriminative training using non-uniform criteria for cross-layer acoustic modeling," in *Proc. ICASSP2012*, pp. 4089–4092, 2012.

[44] C. Weng, B.-H. Juang, and D. Povey, "Discriminative training using non-uniform criteria for keyword spotting on spontaneous speech," in *Proc. InterSpeech2012*, 2012.

[45] C. Weng and B.-H. Juang, "Adaptive boosted non-uniform minimum classification error for keyword spotting," in *submitted to ICASSP2013*, 2013.

[46] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proc. EuroCOLT95*, pp. 23–27, 1995.

[47] R. Schluter, W. Macherey, B. Muller, and H. Ney, "Comparision of discriminative training criteria and optimization methods for speech recognition," *Speech Communications*, vol. 34, pp. 287–310, 2001.

[48] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: Word error minimization and other applications of confusion networks," 2000.

[49] V. Goel and W. J. Byrne, "Minimum bayes-risk automatic speech recognition," *Computer Speech and Language*, vol. 14, no. 2, pp. 115 – 135, 2000.

[50] J. L. B. Zadrozny and N. Abe, "Cost-sensitive learning by cost-proportionate example weighting," in *Proc. ICDM 2003*, pp. 435–442, 2003.

[51] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiat, S. Kombrink, P. Motlicek, Y. Qian, K. Riedhammer, K. Vesely, and N. T. Vu, "Generating exact lattices in the wfst framework," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 4213–4216, March 2012.

[52] P. Gopalakrishnan, D. Kanevsky, A. Nadas, and D.Nahamoo, "Generalization of the baum algorithm to rational objective functions," in *Proc. ICASSP1989*, pp. 631–634, 1989.

[53] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," 1996.

[54] R. Zhang and E. I. Rudnicky, "Comparative study of boosting and non-boosting training for constructing ensembles of acoustic models," in *Proc. EuroSpeech2003*, 2003.

[55] R. Zhang and E. I. Rudnicky, "A frame level boosting training scheme for acoustic modelling," in *Proc. of ICSLP 2004*, 2004.

[56] C. Dimitrakakis and S. Bengio, "Boosting hmms with an application to speech recognition," in *Proc. ICASSP2004*, 2004.

[57] G. Saon and H. Soltau, "Boosting systems for large vocabulary continuous speech recognition," *Speech Communications*, vol. 54, pp. 212–218, Feb. 2012.

[58] J. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," in *Proc. IEEE International Workshop on Automatic Speech Recognition and Understanding.*, pp. 347–354, 1997.

[59] M. Weintraub, "Keyword-spotting using sri's decipher large-vocabulary speech-recognition system," in *Proc. ICASSP 1993*, pp. 463–466, 1993.

[60] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Signal Processing Society, Dec. 2011. IEEE Catalog No.: CFP11SRW-USB.

[61] M. Gales, "Semi-tied covariance matrices for hidden markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 7, pp. 272–281, 1999.

[62] C. Leggetter and P. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Computer Speech and Language*, vol. 9, pp. 171–185, 1995.

[63] F. Pascale, C. Y. Ma, and W. K. Liu, "Map-based cross-language adaptation augmented by linguistic knowledge: From english to chinese," in *In EUROSPEECH*, pp. 871–874, 1999.

[64] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Commun.*, vol. 50, no. 5, pp. 434–451, 2008.

[65] W. Gao, K.-F. Wong, and W. Lam, "Phoneme-based transliteration of foreign names for oov problem," in *Proceedings of the First international joint conference on Natural Language Processing*, pp. 110–119, 2005.

[66] M.-Y. Hwang and X. L. et al., "Progress on mandarin conversational telephone speech recognition," in *in International Symposium on Chinese Spoken Language Processing*, 2004.

[67] A. Acero, "Environmental robustness in automatic speech recognition," in *ICASSP*, IEEE, 1990.

[68] L. Deng, A. Acero, J. Li, and J. Droppo, "High-performance robust speech recognition using stereo training data," in *ICASSP*, IEEE, 2001.

[69] D. Yu, L. Deng, J. Droppo, J. Wu, Y. Gong, and A. Acero, "A minimum-mean-square-error noise reduction algorithm on mel-frequency cepstra for robust speech recognition.," in *ICASSP*, pp. 4041–4044, IEEE, 2008.

[70] H. Hermansky and N. Morgan, "RASTA processing of speech," *Speech and Audio Processing, IEEE Transactions on*, vol. 2, no. 4, pp. 578–589, 1994.

[71] M. J. Hunt and C. Lefebvre, "A comparison of several acoustic representations for speech recognition with degraded and undegraded speech," in *Proc. ICASSP1989*, 1989.

[72] M. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech and Language*, vol. 12, pp. 75–98, 1998.

[73] R. Lippmann, E. Martin, and D. Paul, "Multi-style training for robust isolated-word speech recognition," in *Proc. ICASSP1987*, 1987.

[74] M. Gales, S. Young, and S. J. Young, "Robust continuous speech recognition using parallel model combination," *IEEE Transactions on Speech and Audio Processing*, vol. 4, pp. 352–359, 1996.

[75] P. J. Moreno, *Speech Recognition in Noisy Environments*. PhD thesis, ECE Department, CMU, 1996.

[76] J. Li, L. Deng, D. Yu, Y. Gong, and A. Acero, "High-performance hmm adaptation with joint compensation of additive and convolutive distortions via vector taylor series.," in *ASRU*, pp. 65–70, IEEE, 2007.

[77] Y. Zhao and B.-H. Juang, "On noise estimation for robust speech recognition using vector taylor series," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 4290–4293, 2010.

[78] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, pp. 30 –42, jan. 2012.

[79] D. Yu, M. L. Seltzer, J. Li, J.-T. Huang, and F. Seide, "Feature learning in deep neural networks - a study on speech recognition tasks," *CoRR*, vol. abs/1301.3605, 2013.

[80] M. L. Seltzer, D. Yu, and Y.-Q. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. ICASSP2013*, 2013.

[81] O. Vinyals, S. Ravuri, and D. Povey, "Revisiting Recurrent Neural Networks for Robust ASR," in *ICASSP*, 2012.

[82] A. Maas, Q. Le, T. O'Neil, O. Vinyals, P. Nguyen, and A. Ng, "Recurrent neural networks for noise reduction in robust ASR," in *Proceedings of INTERSPEECH*, 2012.

[83] F. Weninger, J. Geiger, M. Wöllmer, B. Schuller, and G. Rigoll, "The Munich 2011 CHiME Challenge Contribution: NMF-BLSTM speech enhancement and recognition for reverberated multisource environments," in *Proc. Machine Listening in Multisource Environments (CHiME 2011), satellite workshop of Interspeech 2011, ISCA, Florence, Italy*, 2011.

[84] A. Graves, A. Mohamed, and G. Hinton., "Speech recognition with deep recurrent neural networks," in *Acoustics Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013.

[85] E. Vincent, J. Barker, S. Watanabe, J. Le Roux, F. Nesta, and M. Matassoni, "The second 'CHiME' Speech Separation and Recognition Challenge: Datasets, tasks and baselines," in *ICASSP*, (Vancouver, Canada), 2013.

[86] Y. Tachioka, S. Watanabe, J. Le Roux, and J. R. Hershey, "Discriminative methods for noise robust speech recognition: A CHiME challenge benchmark," in *Proceedings of the CHiME 2013 International Workshop on Machine Listening in Multisource Environments*, 2013.

[87] B. Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *ICASSP*, pp. 3761–3764, 2009.

[88] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proceedings of INTERSPEECH*, 2013.

[89] M. Cooke, J. R. Hershey, and S. J. Rennie, "Monaural speech separation and recognition challenge.," *Computer Speech and Language*, vol. 24, no. 1, pp. 1–15, 2010.

[90] R. J. Weiss and D. P. W. Ellis, "Monaural Speech Separation Using Source-Adapted Models," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 114–117, 2007.

[91] Z. Ghahramani and M. I. Jordan, "Factorial hidden markov models," *Mach. Learn.*, vol. 29, pp. 245–273, Nov. 1997.

[92] J. Barker, N. Ma, A. Coy, and M. Cooke, "Speech fragment decoding techniques for simultaneous speaker identification and speech recognition," *Comput. Speech Lang.*, vol. 24, pp. 94–111, Jan. 2010.

[93] J. Ming, T. J. Hazen, and J. R. Glass, "Combining missing-feature theory, speech enhancement and speaker-dependent/-independent modeling for speech separation.," in *INTERSPEECH*, ISCA, 2006.

[94] Y. Shao, S. Srinivasan, Z. Jin, and D. Wang, "A computational auditory scene analysis system for speech segregation and robust speech recognition.," *Computer Speech and Language*, vol. 24, no. 1, pp. 77–93, 2010.

[95] M. N. Schmidt and R. K. Olsson, "Single-channel speech separation using sparse non-negative matrix factorization," in *Interspeech*, sep 2006.

[96] M. R. Every and P. J. B. Jackson, "Enhancement of harmonic content of speech based on a dynamic programming pitch tracking algorithm.," in *INTERSPEECH*, 2006.

[97] J. Li, D. Yu, J.-T. Huang, and Y. Gong, "Improving wideband speech recognition using mixed-bandwidth training data in cd-dnn-hmm.," in *SLT*, pp. 131–136, IEEE, 2012.

[98] V. Abrash, H. Franco, A. Sankar, and M. Cohen, "Connectionist speaker normalization and adaptation," in *in Eurospeech*, pp. 2183–2186, 1995.

[99] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *ASRU*, pp. 24–29, 2011.

[100] D. Yu and M. L. Seltzer, "Improved bottleneck features using pretrained deep neural networks," in *INTERSPEECH*, pp. 237–240, 2011.

[101] T. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 4153–4156, March 2012.

[102] C. Weng, D. Yu, S. Watanabe, and B.-H. Juang, "Recurrent deep neural networks for robust speech recognition," in *ICASSP*, (Florence, Italy), 2014.

[103] J. Geiger, F. Weninger, J. Gemmeke, M. Wollmer, B. Schuller, and G. Rigoll, "Memory-enhanced neural networks and nmf for robust asr," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 22, pp. 1037–1046, June 2014.

[104] A. Graves, N. Jaitly, and A. rahman Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *ASRU*, pp. 273–278, 2013.

[105] M. Cooke, J. Barker, S. Cunningham, and X. Shao, "An audio-visual corpus for speech perception and automatic speech recognition," *The Journal of the Acoustical Society of America*, vol. 120, pp. 2421–2424, November 2006.

[106] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, pp. 14 –22, jan. 2012.

[107] E. M. Grais, M. U. Sen, and H. Erdogan, "Deep neural networks for single channel source separation," in *ICASSP*, (Florence, Italy), 2014.

[108] J. H. Wright, M. J. Carey, and E. S. Parris, "Improved topic spotting through statistical modelling of keyword dependencies," in *Proc. ICASSP1995*, pp. 313–316, 1995.

[109] A. L. Gorin, G. Riccardi, and J. H. Wright, "Automatic acquisition of sailent grammar fragments for call-type classification," in *Proc. EuroSpeech97*, 1997.

[110] K. Myers, M. Kearns, S. Singh, and M. A. Walker, "A boosting approach to topic spotting on subdialogues," in *Proc. ICML00*, pp. 662–669, 2000.

[111] J. Foote, "An overview of audio information retrieval," *Multimedia Syst.*, vol. 7, no. 1, pp. 2–10, 1999.

[112] T. J. Hazen, F. Richardson, and A. Margolis, "Topic identification from audio recordings using word and phone recognition lattices," in *ASRU*, pp. 659–664, 2007.

[113] C. Cortes, P. Haffner, and M. Mohri, "Rational kernels: Theory and algorithms," *Journal of Machine Learning Research*, vol. 5, pp. 1035–1062, 2004.

[114] C. Weng and B.-H. Juang, "Latent semantic rational kernels for topic spotting on spontaneous speech," in *Proc. ICASSP2013*, 2013.

[115] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[116] T. Hofmann, "Probabilistic latent semantic analysis," in *In Proc. of Uncertainty in Artificial Intelligence, UAI'99*, pp. 289–296, 1999.

[117] C. Cortes, P. Haffner, and M. Mohri, "Lattice kernels for spoken dialog classification," in *Proc. ICASSP03*, pp. 628–631, 2003.

[118] N. Cristianini, J. Shawe-Taylor, and H. Lodhi, "Latent semantic kernels," *Journal of Intellignet Information Systems*, vol. 18, no. 2-3, pp. 127–152, 2002.

[119] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," in *INFORMATION PROCESSING AND MANAGEMENT*, pp. 513–523, 1988.

[120] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, "Wordnet: An online lexical database," *International Journal of Lexicography*, vol. 3, pp. 235–244, 1990.

[121] T. Pedersen, S. Patwardhan, and J. Michelizzi, "Wordnet::similarity: measuring the relatedness of concepts," in *Demonstration Papers at HLT-NAACL 2004*, HLT-NAACL–Demonstrations '04, pp. 38–41, 2004.

[122] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, pp. 613–620, Nov. 1975.

[123] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 448–453, 1995.

[124] D. Lin, "An information-theoretic definition of similarity," in *In Proceedings of the 15th International Conference on Machine Learning*, pp. 296–304, Morgan Kaufmann, 1998.

[125] J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," *CoRR*, vol. cmp-lg/9709008, 1997.

[126] C. Leacock and M. Chodorow, *WordNet: An electronic lexical database*. MIT Press, 1972.

[127] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, (Stroudsburg, PA, USA), pp. 133–138, Association for Computational Linguistics, 1994.

[128] Y. Li, D. Mclean, Z. Bandar, J. O'Shea, and K. Crockett, "Sentence similarity based on semantic nets and corpus statistics," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 18, pp. 1138–1150, Aug 2006.

[129] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, (New York, NY, USA), pp. 50–57, ACM, 1999.

[130] T. Hofmann, "Learning the similarity of documents: An information-geometric approach to document retrieval and categorization," in *NIPS*, pp. 914–920, 1999.

[131] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *In Advances in Neural Information Processing Systems 11*, pp. 487–493, MIT Press, 1998.

[132] T. Griffiths, "Gibbs sampling in the generative model of latent dirichlet allocation," tech. rep., Stanford University, 2002.

[133] X. Wei and W. B. Croft, "LDA-based document models for ad-hoc retrieval," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, (New York, NY, USA), pp. 178–185, ACM, 2006.

[134] E. Gaussier and C. Goutte, "Relation between plsa and nmf and implications," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, (New York, NY, USA), pp. 601–602, ACM, 2005.

[135] B. Kingsbury, T. N. Sainath, and H. Soltau, "Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization," in *INTERSPEECH*, 2012.

[136] T. Mikolov, W. tau Yih, and G. Zweig, "Linguistic regularities in continuous space word representations.," in *HLT-NAACL*, pp. 746–751, The Association for Computational Linguistics, 2013.

[137] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, pp. 3111–3119, 2013.

[138] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[139] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Cernocký, "Strategies for training large scale neural network language models," in *ASRU*, pp. 196–201, 2011.

[140] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. W. Senior, P. A. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *NIPS*, pp. 1232–1240, 2012.

[141] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, and G. Zweig, "fMPE: Discriminatively trained features for speech recognition," in *ICASSP (1)*, pp. 961–964, 2005.

[142] J. Martens, "Deep learning via Hessian-free optimization," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, (Haifa, Israel), pp. 735–742, Omnipress, June 2010.

[143] F. Niu, B. Recht, C. Re, and S. J. Wright, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent," *CoRR*, vol. abs/1106.5730, 2011.